

Lecture Notes in Artificial Intelligence 4509

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

VISIT...

LANZAROTE  
*Caliente*.COM

Ziad Kobti Dan Wu (Eds.)

# Advances in Artificial Intelligence

20th Conference of the Canadian Society  
for Computational Studies of Intelligence, Canadian AI 2007  
Montreal, Canada, May 28-30, 2007  
Proceedings

## Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA  
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

## Volume Editors

Ziad Kobti

Dan Wu

University of Windsor

School of Computer Science

401 Sunset Avenue, Windsor, Ontario, N9B 3P4, Canada

E-mail: {kobti,danwu}@uwindsor.ca

Library of Congress Control Number: 2007926815

CR Subject Classification (1998): I.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743

ISBN-10 3-540-72664-0 Springer Berlin Heidelberg New York

ISBN-13 978-3-540-72664-7 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

[springer.com](http://springer.com)

© Springer-Verlag Berlin Heidelberg 2007

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 12066940 06/3180 5 4 3 2 1 0

# Preface

This volume contains the papers presented at AI 2007, the 20th conference of the Canadian Society for the Computational Study of Intelligence (CSCSI). AI 2007 attracted a new record of 260 paper submissions. Each paper was assigned to three reviewers who tirelessly worked to provide high-quality reviews. Out of these, 46 high-quality papers were accepted for publication by the Program Committee. The organization of AI 2007 has benefited from the collaboration of many individuals. Foremost, we express our appreciation to the Program Committee members and the additional reviewers who provided thorough and timely reviews. We are grateful to Andrei Voronkov and the support team assisting with the EasyChair Conference System that hosted the AI 2007 paper submission and review process. We also extend our thanks to the School of Computer Science at the University of Windsor for hosting the conference Web site. Finally, we thank the Organizing Committee and the members of the CSCSI Executive Committee for all their efforts in making AI 2007 a successful conference.

May 2007

Ziad Kobti  
Dan Wu

# Organization

AI 2007 was organized by the Canadian Society for the Computational Study of Intelligence (CSCSI).

## Executive Committee

General Chair	Tal Arbel (McGill University)
Program Co-chairs	Ziad Kobti and Dan Wu (University of Windsor)

## Program Committee

## Referees

Raza Abidi (Dalhousie University)	Nafaa Jabeur (University of Windsor)
Imran Ahmad (University of Windsor)	Nathalie Japkowicz (Monash University)
Esma Aïmeur (University of Montreal)	Grigoris Karakoulas (University of Toronto)
Caroline Barrière (NRC)	Kamran Karimi (University of Lakehead)
Shai Ben-David (University of Waterloo)	Vlado Keselj (Dalhousie University)
Sabine Bergler (Concordia University)	Iluju Kiringa (University of Ottawa)
Cory Butz (University of Regina)	Ziad Kobti (University of Windsor)
Giuseppe Carenini (UBC)	Greg Kondrak (University of Alberta)
Yllias Chali (University of Lethbridge)	Leila Kosseim (Concordia University)
David Chiu (University of Guelph)	Luc Lamontagne (Laval University)
Douglas Dankel (University of Florida)	Philippe Langlais (University of Montreal)
Atilla Elçi (E. Mediterranean University)	Guy Lapalme (University of Montreal)
Michael Fleming (University of New Brunswick)	Kate Larson (University of Waterloo)
Richard Frost (University of Windsor)	François Laviolette (Laval University)
Scott Goodwin (University of Windsor)	Pawan Lingras (Saint Mary's University)
Robin Gras (University of Windsor)	Alejandro Lopez-Ortiz (University of Waterloo)
Jim Greer (University of Saskatchewan)	Choh Man Teng (IHMC)
Adlane Habed (University of Windsor)	Jean-Marc Mercantini (LSIS)
Malcolm Heywood (Dalhousie University)	Joel Martin (NRC)
Robert Hilderman (University of Regina)	Bob Mercer (University of Western Ontario)
Graeme Hirst (University of Toronto)	
Diana Inkpen (University of Ottawa)	

David Nadeau (NRC)	Ahmed Tawfik (University of Windsor)
Eric Neufeld (University of Saskatchewan)	Nicole Tourigny (Laval University)
Roger Nkambou (University of Montreal)	Thomas Tran (University of Ottawa)
David Poole (UBC)	Andre Trudel (Acadia University)
Jian-Yun Nie (University of Montreal)	Marcel Turcotte (University of Ottawa)
Gerald Penn (University of Toronto)	Peter van Beek (University of Waterloo)
Fred Popowich (Simon Fraser University)	Herna Viktor (University of Ottawa)
Robert Reynolds (Wayne State University)	Shaojun Wang (Wright State University)
Massih Reza Amini (Pierre and Marie Curie University)	Dan Wu (University of Windsor)
Luis Rueda (University of Concepcion)	Yang Xiang (University of Guelph)
Anoop Sarkar (Simon Fraser University)	Yiyu Yao (University of Regina)
Weiming Shen (NRC)	Jia You (University of Alberta)
Michel Simard (NRC)	Li-Yan Yuan (University of Alberta)
Stan Szpakowicz (University of Ottawa)	Harry Zhang (University of New Brunswick)
	Nur Zincir-Heywood (Dalhousie University)

## Additional Reviewers

### Referees

Tony Abou-Assaleh	Baohua Gu	Behnam Rahnema
Dulce Aguilar-Solis	Franklin Hanshar	Maxim Roy
Muath Alzghool	Michael Horsch	Elhadi Shakshuki
Xiangdong An	Zina Ibrahim	Danny Silver
Alina Andreevskaia	Michael Janzen	Zhongmin Shi
Jing Bai	Sittichai Jiampojarn	Jiang Su
Susan Bartlett	Mehdi M. Kashani	Hathai Tanta-ngai
Yaohua Chen	Abolfazl	Jeff Taylor
William Elazmeh	Keighobadi Lamjiri	Javier Thaine
Mark Eramian	Alistair Kennedy	Davide Turcato
Oana Frunza	Guohua Liu	Bin Wang
Sébastien Gambs	Haibin Liu	Hong Yao
Liqiang Geng	Mehrdad Oveisi-Fordoei	Haiyi Zhang
Kevin Grant	Robert Price	

### Sponsoring Institutions

The Canadian Society for the Computational Study of Intelligence (CSCSI)  
*Société Canadienne pour L'Étude de l'Intelligence par Ordinateur*  
 Ontario Centres of Excellence (OCE)

# Table of Contents

## Session 1. Agents

Modeling Role-Based Agent Team.....	1
<i>Yu Zhang</i>	
Distributed Collaborative Filtering for Robust Recommendations Against Shilling Attacks .....	14
<i>Ae-Ttie Ji, Cheol Yeon, Heung-Nam Kim, and Geun-Sik Jo</i>	
Competition and Coordination in Stochastic Games .....	26
<i>Andriy Burkov, Abdeslam Boularias, and Brahim Chaib-draa</i>	
Multiagent-Based Dynamic Deployment Planning in RTLS-Enabled Automotive Shipment Yard .....	38
<i>Jindae Kim, Changsoo Ok, Soundar R.T. Kumara, and Shang-Tae Yee</i>	
R-FRTDP: A Real-Time DP Algorithm with Tight Bounds for a Stochastic Resource Allocation Problem .....	50
<i>Camille Besse, Pierrick Plamondon, and Brahim Chaib-draa</i>	
A Reorganization Strategy to Build Fault-Tolerant Multi-Agent Systems .....	61
<i>Sehl Mellouli</i>	

## Session 2. Bioinformatics

A Multi-site Subcellular Localizer for Fungal Proteins .....	73
<i>Michel Nathan</i>	
Selecting Genotyping Oligo Probes Via Logical Analysis of Data .....	86
<i>Kwangsoo Kim and Hong Seo Ryoo</i>	

## Session 3. Classification

Learning the Semantic Meaning of a Concept from the Web .....	98
<i>Yang Yu and Yun Peng</i>	
On Combining Dissimilarity-Based Classifiers to Solve the Small Sample Size Problem for Appearance-Based Face Recognition .....	110
<i>Sang-Woon Kim and Robert P.W. Duin</i>	
A Novel Approach for Automatic Palmprint Recognition .....	122
<i>Murat Ekinci and Murat Aykut</i>	



ICS: An Interactive Classification System .....	134
<i>Yan Zhao, Yiyu Yao, and Mingwu Yan</i>	
Fast Most Similar Neighbor Classifier for Mixed Data .....	146
<i>Selene Hernández-Rodríguez, J. Francisco Martínez-Trinidad, and J. Ariel Carrasco-Ochoa</i>	
Performance Measures in Classification of Human Communications .....	159
<i>Marina Sokolova and Guy Lapalme</i>	
Cost-Sensitive Decision Trees with Pre-pruning .....	171
<i>Jun Du, Zhihua Cai, and Charles X. Ling</i>	
Probability Based Metrics for Locally Weighted Naive Bayes .....	180
<i>Bin Wang and Harry Zhang</i>	
Recurrent Boosting for Classification of Natural and Synthetic Time-Series Data .....	192
<i>Robert D. Vincent, Joelle Pineau, Philip de Guzman, and Massimo Avoli</i>	
Pattern Classification in No-Limit Poker: A Head-Start Evolutionary Approach .....	204
<i>Brien Beattie, Garrett Nicolai, David Gerhard, and Robert J. Hilderman</i>	

## Session 4. Constraint Satisfaction

Managing Conditional and Composite CSPs .....	216
<i>Malek Mouhoub and Amrudee Sukpan</i>	
Multiagent Constraint Satisfaction with Multiply Sectioned Constraint Networks .....	228
<i>Yang Xiang and Wanling Zhang</i>	

## Session 5. Data Mining

A Clustering Algorithm Based on Adaptive Subcluster Merging .....	241
<i>Jiani Hu, Weihong Deng, and Jun Guo</i>	
Efficient Algorithms for Video Association Mining .....	250
<i>B. SivaSelvan and N.P. Gopalan</i>	
Distributed Data Mining in a Ubiquitous Healthcare Framework .....	261
<i>Murlikrishna Viswanathan</i>	
Constructing a User Preference Ontology for Anti-spam Mail Systems .....	272
<i>Jongwan Kim, Dejing Dou, Haishan Liu, and Donghui Kwak</i>	

Question Answering Summarization of Multiple Biomedical Documents .....	284
<i>Zhongmin Shi, Gabor Melli, Yang Wang, Yudong Liu, Baohua Gu, Mehdi M. Kashani, Anoop Sarkar, and Fred Popowich</i>	

A Profit-Based Business Model for Evaluating Rule Interestingness .....	296
<i>Yaohua Chen, Yan Zhao, and Yiyu Yao</i>	

## Session 6. Knowledge Representation and Reasoning

Reasoning About Operations on Sets .....	308
<i>Bernhard Heinemann</i>	

Analytic Results on the Hodgkin-Huxley Neural Network: Spikes Annihilation .....	320
<i>Dragos Calitoiu, John B. Oommen, and Doron Nussbaum</i>	

Improving Importance Sampling by Adaptive Split-Rejection Control in Bayesian Networks .....	332
<i>Changhe Yuan and Marek J. Druzdzel</i>	

Adding Local Constraints to Bayesian Networks .....	344
<i>Mark Crowley, Brent Boerlage, and David Poole</i>	

On the Use of Possibilistic Bases for Local Computations in Product-Based Possibilistic Networks .....	356
<i>Salem Benferhat and Salma Smaoui</i>	

Reasoning with Conditional Preferences Across Attributes .....	369
<i>Shaoju Chen, Scott Buffett, and Michael W. Fleming</i>	

Path Propagation for Inference in Bayesian Networks .....	381
<i>Dan Wu and Liu He</i>	

Problem-Solving Knowledge Mining from Users' Actions in an Intelligent Tutoring System .....	393
<i>Roger Nkambou, Engelbert Mephu Nguifo, Olivier Couturier, and Philippe Fournier-Viger</i>	

Incremental Neighborhood Graphs Construction for Multidimensional Databases Indexing .....	405
<i>Hakim Hacid and Tetsuya Yoshida</i>	

## Session 7. Learning

Learning Network Topology from Simple Sensor Data .....	417
<i>Dimitri Marinakis, Philippe Giguère, and Gregory Dudek</i>	

Reinforcement Learning in Nonstationary Environment Navigation Tasks .....	429
<i>Terran Lane, Martin Ridens, and Scott Stevens</i>	

On the Stability and Bias-Variance Analysis of Kernel Matrix Learning .....	441
<i>V. Vijaya Saradhi and Harish Karnick</i>	

## Session 8. Natural Language

Query-Based Summarization of Customer Reviews .....	452
<i>Olga Feiguina and Guy Lapalme</i>	

Multi-state Directed Acyclic Graphs .....	464
<i>Michael Wachter and Rolf Haenni</i>	

Fuzzy Clustering for Topic Analysis and Summarization of Document Collections .....	476
<i>René Witte and Sabine Bergler</i>	

Creating a Fuzzy Believer to Model Human Newspaper Readers .....	489
<i>Ralf Krestel, René Witte, and Sabine Bergler</i>	

Rethinking the Semantics of Complex Nominals .....	502
<i>Nabil Abdullah and Richard A. Frost</i>	

A Hybrid Approach to Improving Automatic Speech Recognition Via NLP .....	514
<i>Kimberly Voll</i>	

## Session 9. Planning

Planning in Multiagent Expedition with Collaborative Design Networks .....	526
<i>Yang Xiang and Frank Hanshar</i>	

Hierarchical Shortest Pathfinding Applied to Route-Planning for Wheelchair Users .....	539
<i>Suling Yang and Alan K. Mackworth</i>	

<b>Author Index</b> .....	551
---------------------------	-----

# Modeling Role-Based Agent Team\*

Yu Zhang

Computer Science Department, Trinity University, San Antonio, TX 78212  
Tel.:(210)999-7399, Fax:(210)999-7477  
yzhang@cs.trinity.edu

**Abstract.** The problem of ensuring agents work as an effective team in dynamic distributed environments still remains a challenging issue. In this paper we proposed a role-based team model. In our model, the role characterizes the responsibilities and provides logic patterns to achieve certain goals and cooperate with others. The agent is an autonomous execution unit and follows the logic patterns that the role provides. We also developed algorithms and mechanisms to evolve the plan of a role to the plan of an agent. Our role-based team model allows the split of roles (who define the plans) and agents (who execute the plans) in team plans, and dynamic role-agent assignment. It also achieves a certain level of plan reusability. We present two experiments which show plan reusability and its flexibility in supporting simultaneously plan invocation.

**Keywords:** Agent teamwork, Role, Plan.

## 1 Introduction

Teamwork is becoming increasingly important in many dynamic multi-agent systems [13]. Agents in a team need to form joint mental states which drive agents to act together as a team and form the interactions leading their individual actions to team efforts [5, 7]. To simulate teamwork, a teamwork language is demanded to explicitly express the mental states underlying teamwork. In our opinion, the effective design of a teamwork language requires two aspects to consider. First, it should be able to handle unexpected uncertainties occurred in complex and dynamic domains, such as dynamic changes in team's goals, team members' unexpected failures to fulfill their responsibilities, decision-making in dynamic environment, and dynamically backing up other team members. Second, considering the perspective of software engineering, the teamwork language would better allow specify teamwork knowledge conceptually for being reused, particularly, team plans are better specified in terms of abstract entities, instead of specific agents, so as to be reused by different teams of agents.

A lower level of abstraction, role, is currently used by many researcher of multi-agent systems to close this gap [14, 9, 11, 6, 16]. Biddle and Thomas's role theory views role as the concept of partitioning behaviors and emphasizing coordination and cooperation [2]. Becht's ROPE (Role Oriented Programming Environment for multi-agent systems) uses roles to decouple the organization of agents from the structure of cooperation processes [1]. Cooperation process is designed from a global perspective

---

\* This work was supported by DoD MURI F49620-00-I-326 administered through AFOSR.

and largely independent of concrete agents so that shifting cooperative behavior does not require changing agents (agents can fill multiple roles and switch between them). Stone and Veloso introduced roles as a mechanism for specifying an agent's internal and external behaviors and decomposing team tasks [12]; they then used this to model robot soccer. A formation decomposes the task space by defining a set of roles. There are as many roles as there are agents in the team, so that each role is filled by one agent. The mapping between agents and role is not pre-specified.

In this paper, we propose a role-based teamwork language. Different with the existing work, we use roles and role variables distinguish static (by roles) and dynamic (by role variables) action associations; and when delegating roles and role variables in a plan to agents, we have the agents form a joint mental state to enforce the execution of the plan as a team effort, particularly the sub-actions in the plan will be executed coherently. Our concepts of role and role variable enable our mechanisms of task decomposition and delegation, by which role-based plans drive agents to actually execute teamwork. Our mechanism of task decomposition is based on a notion of responsibility, which is defined in terms of what a responsibility contains and how a responsibility impacts the mental states of the agent(s) taking the responsibility. Our mechanism of task delegation has three steps: 1) a team task is translated to a team responsibility which is represented by a graph; 2) through decomposing the team responsibility graph to individual responsibility graphs, a team task is decomposed to individual sub-tasks; and 3) individual sub-tasks are delegated to agents.

The structure of this paper is as follows. Section 2 formalizes the basic concepts. Section 3 introduces algorithms for task decomposition and delegation. Section 4 introduces delegating roles to agents, formalize a notion of admissible assignment, and present a CSP algorithm to search for admissible assignments. Section 5 is experiment. Section 6 concludes the paper.

## 2 Role-Based Plan

In their role theory, Biddle and Thomas concluded that roles can be defined based on partitioning concepts for persons and their behaviors [2]. They also pointed out that pre-association with roles is too restrictive, i.e. the determination of which role actually performs which action being determined dynamically in a specific situation. Based on it, we define a position as all entities that can perform a set of primitive operations, and use this to characterize a collectively recognized category of persons who are able to exhibit a set of behaviors. We define a role as an entity of a position associated with a bag of temporally ordered actions. We define a role variable as an entity which is dynamically selected from a set of roles to be associated with a bag of temporally ordered actions. Generally speaking, a role variable stands for some role out of a set of roles. Depending on concrete situation, one role from the set is dynamically selected to fill the role variable. When this happens, the bag of operations of the assuming role is dynamically expanded.

### 2.1 Position

We define a concept of position to refer a collectively recognized category of entities (persons) that exhibit a set of behaviors.

**Definition 1.** An **entity** is an abstraction of any performer (e.g., agent or person) that is able to perform operations.

**Definition 2.** A **position** is a named set of all entities that are able to perform a set of primitive operations, denoted by operators. Given a set of operations  $O$ , a position based on the operations is

$$\mathcal{A}(O) = \{e | e \in \text{Entities} \wedge \forall o (o \in O) \wedge \text{Capable}(e, o)\}, \quad (1)$$

where  $e$  represents an entity and  $\text{Capable}(e, o)$  means that  $e$  is able to perform operation  $o$ , assuming the preconditions of  $o$  are true.

The purpose of defining position is to capture the capability requirements on agents. Every entity of a position is required to be capable to do the operators defined in the position. For example,

$$(\text{POSITION sniffer} (\text{talk move movein randmove sense selectTarget})). \quad (2)$$

A position *sniffer* is defined by a set of operators, including *talk*, *move*, *movein*, *randmove*, *sense*, and *selectTarget*. Suppose  $e$  is an entity that takes on the position *sniffer*;  $e$  should be able to perform the above operators.

## 2.2 Role

We base our definition for a role on a position. In other words, any action (i.e., primitive operation) associated with the role must be in the operation set of the position. Let  $O$  be a bag of operations that must be performed by the same entity of a position,  $COND$  be a bag of conditional operators where each action is contingent on a conjunction of conditions, and  $C_O$  be a bag of ordering constraints that impose a “temporal order” on  $O$  and the evaluation of the conditions in  $COND$ . We define a role  $r$  as an abstraction of the entity of position  $P$  that satisfies the constraints.

**Definition 3.** A **role** is an abstraction of an entity that performs a specific bag of operations and includes temporal constraints on the order in which the operations may be performed:

$$r = (id, \mathcal{A}, O, COND, \prec_r). \quad (3)$$

where  $id$  is the name of  $r$  and refers to the entity of position  $\mathcal{A}$  that must perform the operations in  $O$ .

$\prec_r$  is a set of temporal orders as  $(o_i, o_j)$  where  $o_i, o_j \in O \cup COND \cup S$ , and  $S = \{o_s, o_e\}$ .  $o_s$  is a dummy starting operation that can be performed by any entity of a position, and  $o_e$  is a dummy ending operations that can be performed by any entity of a position. Temporal orders are transitive. That is, if there are two temporal orders  $(o_i, o_j)$  and  $(o_j, o_k)$ , then  $(o_i, o_k)$  is a true temporal order too. However,  $\prec_r$  is not a transitive closure. That means, even though  $(o_i, o_j)$  and  $(o_j, o_k)$  are in  $\prec_r$ ,  $(o_i, o_k)$  may not be in  $\prec_r$ . And, cycles may exist in the set of temporal orders  $\prec_r$ . That is, there exist temporal orders as  $(o_1, o_2), (o_2, o_3), \dots, (o_k, o_1)$ , where  $k \geq 2$ . This does not mean a temporal conflict. Rather, it means that the operators may be executed more than once in

accordance with the temporal orders. The feature also holds in the temporal orders of role variables (described later in Sec. 2.3).

### 2.3 Role Variable

A role variable is similar to a role except that the role variable is dynamically selected from a set of roles. So, in the definition of a role variable, we do not explicitly require it to belong to any specific position. Rather, we require a selection constraint whose satisfaction will select one role out of a set of roles to perform the operations of the role variable in accordance with the specified order. For clarity, we prefix role variables by “?”.

**Definition 4.** A **role variable** is an abstraction of an entity that is dynamically selected from a set of roles to be associated with a bag of temporally ordered actions:

$$?r = (?id, O, COND, <_n, RS, SC). \quad (4)$$

where  $?id$  is the name of  $?r$ ,  $RS$  is the selection scope, i.e., a set of roles,  $SC$  is the selection constraint.

### 2.4 Role-Based Plan

In RoB-MALLET, processes are the place to specify team activities in terms of roles or role variables.

**Definition 5.** Suppose the **process** of a role-based plan is composed by a set of roles  $r_1, r_2, \dots, r_m$ , where  $1 \leq i \leq m$ , and a set of role variables  $?r_1, ?r_2, \dots, ?r_n$ , where  $1 \leq j \leq n$ . The process, denoted *Proc*, is the union of the roles and role variables:

$$Proc = \bigcup_{i=1}^m O_{r_i} \cup \bigcup_{j=1}^n O_{?r_j}, \bigcup_{i=1}^m Cond_{r_i} \cup \bigcup_{j=1}^n Cond_{?r_j}, \bigcup_{i=1}^m <_{r_i} \cup \bigcup_{j=1}^n <_{?r_j}, \bigcup_{j=1}^n RS_{?r_j}, \bigcup_{j=1}^n SC_{?r_j} \quad (5)$$

The difference between the actions associated with roles and actions associated with role variables is that, action associations by roles are static (or direct) while action associations by role variables are dynamic (or indirect). In this way, the process of a role-based plan allows both static and dynamic action association. We call the aggregation of the roles in a role-based plan the *virtual team* of the plan, denoted by *VT*. We call the aggregation of the role variables in a role-based plan as role variable set of the plan, denoted by *RVS*. We note that, roles in *VT* may form a sub-team to invoke a sub-plan and the sub-plan contains a virtual team too.

To delegate roles to agents, the agents must be “qualified” to fill the roles (we call this process role assignment, see Sec. 4 for detail). A necessary condition for this is that the agent must be able to perform operations required by the role. As well as the qualification of the roles, there can be a set of social norms (i.e., conditions), which specify the selection of agents for roles. The social norms may exist for different kinds of reasons, such as geographical reasons, workload balance, cooperation requirement. We define social norms as a set of constraints on delegating roles to agents, and denote these constraints by  $\Gamma$ .

**Definition 6.** A **role-based plan** is defined as

$$P = (\Phi, \theta, \Psi, VT, \Gamma, RVS, Proc) \quad (6)$$

where  $\Phi$  is a set of preconditions,  $\theta$  is a set of effects,  $\Psi$  is a set of termination conditions,  $VT$  is a virtual team,  $\Gamma$  is a set of constraints specifying social norms on delegating roles to agents,  $RVS$  is a set of role variables to specify non-predetermined actions, and  $Proc$  is the process of actions of  $VT$  achieving the pursuing goal (i.e., effects) under similar situations (i.e., preconditions). By this means, team plans are reusable for different agents, i.e. they are to achieve the effects in any situation in which the preconditions are satisfied, independent of the agents executing the plan.

Figure 1 shows an example role-based plan. It is a multi-agent version of the classic wumpus world problem [11]. It defines a role-based plan `scanandkill`. The virtual team consists of three roles,  $r_1$ , a **sniffer**, and  $r_2$  and  $r_3$ , **fighters**. Also, a role variable  $?fi$  is selected from  $r_2$  and  $r_3$  and the one closest to the wumpus to be killed is selected. The constraints on delegating agents to roles also are specified. To fill  $r_2$  and  $r_3$ , agents must have arrows, and  $r_2$  and  $r_3$  must be filled by different agents.

```
(position sniffer (move sense selectTarget))
(position fighter (move shoot))
(plan scanandkill ()
  (Roleteam ((role (r1) sniffer) (role (r2 r3) fighter)))
  (Constraint (hasarrow r2) (hasarrow r3) (NE r2 r3))
  (Process
    (par
      (while (cond (eq 1 1))
        (seq
          (do r1 (selectTarget))
          (if (cond (targetloc r1 ?X1 ?Y1))
            (do r1 (senseandmoveto ?X1 ?Y1))))))
      (while (cond (eq 1 1))
        (if (cond (wumpus ?X2 ?Y2))
          (seq
            (select ?fi (r2 r3) (closestto ?X2 ?Y2))
            (do ?fi (moveclosestto ?X2 ?Y2))
            (do ?fi (shoot ?X2 ?Y2)))))))
```

**Fig. 1.** An Example Role-Based Plan

### 3 Responsibility

#### 3.1 Generating Team Responsibility Graph for Role-Based Plan

A responsibility is an obligation by an agent or a team of agents to perform actions, which forms an intention to do the actions. Responsibility is represented as a directed bipartite graph  $G(V, E)$ , where  $V$  is a set of nodes and  $E$  is a set of directed links connecting nodes. The nodes represent the entrance and exit of a plan  $P$ , the evaluation of conditions, operations, cooperation among roles, and flow control. The links represent various kinds of relations among nodes, including dependencies between two nodes, coordination connections, sub-plan connections, goal connections, and connections among roles for the selection of role variables.



---

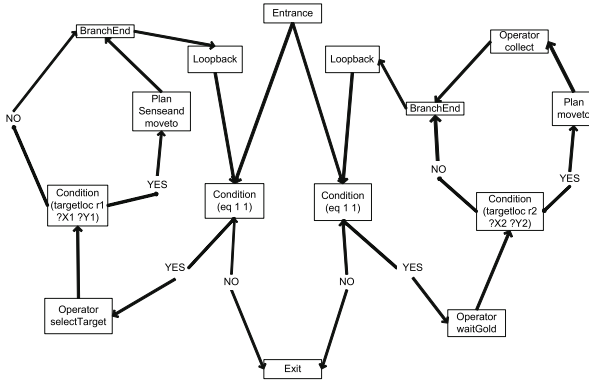
generateTeamResponsibility ( $P$ )

---

- 1: create a team responsibility,  $Resp$ ;
  - 2: build the syntax tree of  $P$ ,  $tree$ ;
  - 3: let  $Proc$  be the root of  $tree$ ;
  - 4: create an entrance node,  $StartNode$ , in  $Resp$ ;
  - 5: create an exit node,  $ExitNode$ , in  $Resp$ ;
  - 6: Translate( $Proc$ ,  $StartNode$ ,  $ExitNode$ ,  $Resp$ );
- 

**Fig. 2.** Generating Team Responsibility Graph for Role-Based Plan  $P$

Figure 2 is the algorithm that translates the process of a role-based plan into a graph of a team responsibility. Figure 3 shows team responsibility graph of the plan `scanandcollect` (see Fig. 1) after reducing redundant connector nodes.



**Fig. 3.** A Reduced Team Responsibility Graph

### 3.2 Decomposing a Team Responsibility Graph to Individual Responsibility Graphs

We decompose a team responsibility graph into individual responsibility graphs by ownerizing the nodes and links in the team responsibility graph to the roles and role variables in the role-based plan. As a result, the individual responsibility graph of a role (or role variable) contains the nodes and links ownerized to the role (or role variable).

The decomposition needs to ensure two things: 1) every action in team responsibility is included into the responsibility of the role (or role variable) with which the action is associated; and 2) the temporal orders on the actions in team responsibility can be preserved so that the joint intention to the actions in the team responsibility can be realized by individual intentions to the actions in the individual responsibilities.

As described in previous Sec. 3.1, when we translate a role-based plan into a team responsibility graph, we translate the actions in the process into corresponding nodes and ownerize the nodes to the roles or role variables with which the actions are associated. If an action is an invocation of an individual operator, the action is translated into an operator node and the operator node is ownerized to the role (or role variable)

which invokes the operator. If an action is an invocation of a team operator, the actions are translated into individual operators corresponding to the team operator and one of the individual operators is ownerized to each involved role (or role variable). If an action is a complex action other than a team operator, such as a role selection, a sub-plan invocation, or a goal achievement, the action is translated to a set of nodes corresponding to the action and one of the nodes is ownerized to each involved role (or role variable).

Because a link connects two nodes and represents the relationship between two nodes, the owner of the link depends on what the owners of the two nodes are. If the owners of the two nodes are the same role or role variable, the owner of the link is consequently the same role or role variable. Otherwise, if the owners of the two nodes are different, the link implies communication between the owners of the two nodes and the source of the link is the initiator of the communication. It is thus very natural to let the owner of the link be that of the source node.

Figure 4 is the algorithm for decomposition. *Resp* is the team responsibility graph of *P* with entrance node, *Start*, and exit node, *End*. The algorithm visits the graph of team responsibility using Depth-First Search from two directions, following the directions of links and starting from the node *Start*, and following the reverse direction of links and starting from the node *End*. An array, *VisitedList*, is used to keep track of visited nodes.

---

decomposeTeamResponsibility (*Resp*, *Start*, *End*)

---

1: create array *visitedList*;  
 2: forwardOwnerize(*Resp*, *Start*);  
 3: reset *visitedList* to empty;  
 4: backwardOwnerize(*Resp*, *End*);

---

**Fig. 4.** Decomposing A Team Responsibility To Individual Responsibility

## 4 Role-Agent Assignment

Our mechanism of task delegation is realized by delegating individual responsibilities to agents. Our mechanism of task delegation contains two levels of responsibility delegations. The first level is delegating the team responsibilities of all the roles in a role-based plan to the agents invoking the role-based plan before the agents start executing the role-based plan. The second level is dynamically delegating the responsibilities of role variables during the execution of the role-based plan. If a role is selected to fill a role variable, the responsibility of the role variable is dynamically included into the responsibility of the role. Given that a certain agent has intended to the responsibility of the role in the first level, the responsibility of the role variable is implicitly and dynamically delegated to the agent. Therefore, key issues in our mechanism of task delegation are the determining a team assignment for invoking a role-based plan and selecting a role for a role variable.

#### 4.1 Role-Agent Assignment and Admissibility

We define a *role-agent assignment* to be a delegation of a role  $r$  to an agent  $ag$ , denoted by  $Assign(r, ag)$ . Once a role  $r$  is delegated to an agent  $ag$ , agent  $ag$  commits to the responsibility of role  $r$ . Consequently, agent  $ag$  intends to perform the actions in the responsibility of role  $r$ , i.e.  $Intend(ag, r, P)$ .

To actually execute the actions in the responsibility of role  $r$ , agent  $ag$  must be capable of the actions. Otherwise, agent  $ag$  is not “qualified” for role  $r$ , or  $Assign(r, ag)$  is not feasible. We use a notion of *admissibility* of role-agent assignment to characterize feasible role-agent assignments.

**Definition 7.**  $Assign(r, ag)$  is **admissible** iff

$$oprequirement(r) \subseteq capability(ag), \quad (7)$$

where  $oprequirement(r)$  is the qualification of role  $r$ , and  $capability(ag)$  is the set of operators of which agent  $ag$  is capable.

#### 4.2 Team Assignment and Admissibility

Suppose that a team of agents  $T$  wants to invoke a role-based plan  $P$  with virtual team  $VT$ . We define a team assignment for  $T$  invoking the role-based plan  $P$  to be the aggregation of role-agent assignments of all roles in  $VT$  to the agents in  $T$ , denoted by  $Assign(VT, T)$ . It could happen that more than one role is delegated to a single agent. Once the roles in  $VT$  are delegated to the agents in  $T$ , the agents jointly commit to the team responsibility of  $VT$ . Consequently, the agents jointly intend to perform the actions in the team responsibilities of  $VT$ , i.e.,  $JIntend(T, VT, Assign(VT, T), P)$ . To perform the actions in the team responsibility of  $VT$ , every agent  $ag$  in  $T$  intends to perform the actions in the responsibility of role  $r$  delegated to the agent  $ag$  according to  $Assign(VT, T)$ , i.e.,  $Intend(ag, r, P)$ .

Similarly, admissibility is required for team assignment.

**Definition 8.**  $Assign(VT, T)$  is **admissible** iff it satisfies the following conditions:

1. For every  $Assign(r, ag)$  in  $Assign(VT, T)$ ,  $Assign(r, ag)$  is admissible;
2. Every constraint in the constraint set of plan  $P$  is satisfied under  $Assign(VT, T)$ ;
3. For every sub-plan invocation, there is an admissible team assignment.

We note that, the above formalism is only to search for an admissible team assignment for  $T$  invoking  $P$ , and it occurs when the execution reach the point when  $T$  invokes  $P$ . The statements inside  $P$ , including condition evaluations and sub-plan invocations, have not been reached yet. Checking condition 3 is to ensure that with the found admissible team assignment for  $T$  invoking  $P$ , there will be admissible team assignments for sub-plan invocations of  $P$  during the future execution of  $P$  (the team assignment searching for sub-plan invocations occurs when the execution reach them).

One may question why not just decide which role is delegated to which agent during the execution of a role-based plan (i.e., at the first moment when a role invokes an

action). The reason why we search for an admissible team assignment before actually executing a role-plan is that, we want to not only decide which role is delegated to which agent, but also have agents form a joint intension to enforce the execution of the role-based plan to be a team effort; moreover, the joint intension ensures that every action in  $P$  will be executed in accordance with the temporal ordering on them unless the termination conditions of the role-based plan are satisfied.

### 4.3 CSP Algorithm of Searching for Admissible Team Assignments

Suppose a team of agents  $T$  wants to invoke a role-based plan,  $P$ . The virtual team VT of plan  $P$  is a set of roles, and the constraints of plan  $P$  is  $C_O$ . In the process of  $P$ , there is a set of sub-plan invocations  $Inv$ . The problem of searching for an admissible team assignment for  $T$  invoking  $P$  can be formalized as a constraint satisfaction problem (CSP) [9]:

- The set of variables of the CSP is VT of plan  $P$ .
- The nonempty domain of the CSP is  $T$ . The possible values for each variable (role) are the agents in  $T$ .
- The set of constraints of the CSP includes three types of constraints: 1) qualification for delegating roles to agents, i.e. for any role  $r_i$  in VT,  $oprequirement(r_i) \subseteq Capability(ag)$ , where  $ag$  is the agent assigned to  $r_i$ ; 2) the constraints in  $C_O$ ; and 3) an admissible team assignment for each sub-plan invocation in  $Inv$ .

An algorithm of backtracking search has been developed to search for an admissible team assignment for agents invoking a role-based plan.

### 4.4 Role Selection

A role selection is the select of a role  $r$  to fill a role variable  $?r$ . By Definition 4, role variable declaration includes the selection scope  $RS$  and the selection constraints  $SC$ . On the other hand, the capability requirement of every role in  $RS$  must include all actions associated with the  $?r$ . However, the capability requirement may not include

---

```

roleSelection (?r, RS, SC)


---


1: loop{
2:   if  $\forall$  roles  $\in RS$  have been tested
3:     return failure;
4:   randomly select an untested role  $r$  from  $RS$ ;
5:   let  $ag$  be the agent to which role  $r$  is delegated;
6:   if  $\forall$  constraints  $\in SC$  are satisfied with  $?r$  substituted by  $ag$ 
7:     return  $r$ ;
8:   mark  $r$  as a tested role;}


---



```

**Fig. 5.** Selecting A Role for A Role Variable

all actions associated with  $?r$ . Correspondently, if role  $r$  is selected to fill  $?r$ , the agent to whom  $r$  delegated may not be capable to do some action(s) associated with  $?r$ . To prevent this oddity, a plan can be statically checked to make sure all role variables in the plan are well defined. Also, the agents to whom the selected roles are delegated must be able to execute the actions associated with the role variables. For this reason, we only consider the selection constraints but not the capability issue when resolving role selections. Figure 5 is the algorithm for role selection.

## 5 Experiments

We choose an extension of the wumpus world described in [10], a multi-agent wumpus world, as the domain for our experiments. There are three roles including  $r_1$ , **sniffer**,  $r_2$ , **fighter**, and  $r_3$ , **carrier**.

**Table 1.** The Durations of Operators

Operator	Duration (ms)
move	200
sense	100
shoot	100
collect	100

Table 1 lists operators and their durations. A **sniffer** can perform **sense**, a **fighter** can perform **shoot**, and a **carrier** can perform **collect**. They all can perform **move**. The goal of the team is to kill wumpuses and collect gold. The team plan **scanandkill** was shown in previous Fig. 1.

We collect three data: 1) the number of wumpuses killed, 2) the amount of gold collected, and 3) time performance by the absolute execution time of a plan.

We report two experiments. The first explores plan reusability of RoB-MALLET. The second shows its flexibility in supporting simultaneity of invocation.

### 5.1 Evaluating Plan Reusability

Experiment 1 evaluates that RoB-MALLET is flexible enough to allow reuse of role-based plans by running a role-based plan with different formation of agents.

Table 2 defines five teams of agents that could invoke a role-based plan in the wumpus world. The randomly generated world is 10×10 squares and contains 20 wumpuses and 20 pieces of gold. Agents initially have no knowledge of the squares.

Table 3 shows that all five teams have successfully completed the teamwork. Even though they have different time performances of accomplishing the teamwork, our model is flexible for different formations of teams to invoke the plan. We note that, each of teams T2, T3, and T4 consists of two agents, however, the two agents in different teams play different roles. The workload of the agent taking  $r_1$  (**sniffer**) directly

**Table 2.** Teams with Different Formations Executing the Plan

Team	Agent	Capability	Role(s)
T1	ag <sub>1</sub>	move, sense, shoot, collect	r <sub>1</sub> , r <sub>2</sub> , r <sub>3</sub>
T2	ag <sub>2</sub>	move, sense	r <sub>1</sub>
	ag <sub>3</sub>	move, shoot, collect	r <sub>2</sub> , r <sub>3</sub>
T3	ag <sub>4</sub>	move, shoot	r <sub>2</sub>
	ag <sub>5</sub>	move, sense, collect	r <sub>1</sub> , r <sub>3</sub>
T4	ag <sub>6</sub>	move, collect	r <sub>3</sub>
	ag <sub>7</sub>	move, sense, shoot	r <sub>1</sub> , r <sub>2</sub>
T5	ag <sub>8</sub>	move, sense	r <sub>1</sub>
	ag <sub>9</sub>	move, shoot	r <sub>2</sub>
	ag <sub>10</sub>	move, collect	r <sub>3</sub>

**Table 3.** The Duration of Different Teams Executing the Plan

Team	Time to kill all wumpuses (s)	Time to collect all gold (s)
T1	387	429
T2	151.5	183
T3	285	241
T4	253	296.5
T5	139.5	163.5

decides team performance. The faster agent finds wumpuses and gold, the faster the team finishes the teamwork. In T2, the agent taking r<sub>1</sub> did not take another role. In T3 and T4, the agents taking r<sub>1</sub> also took another role (r<sub>2</sub> in T3 and r<sub>3</sub> in T4). Consequently, the team performance of T2 is better than T3 and T4. For the similar reasons, T3 killed wumpuses slower than T4 because the agent taking r<sub>2</sub> did not take other role in T3 while the agent taking r<sub>2</sub> took r<sub>1</sub>; and T3 collected gold faster than T4 because the agent taking r<sub>3</sub> did not take other role in T3 while the agent taking r<sub>3</sub> took r<sub>1</sub>.

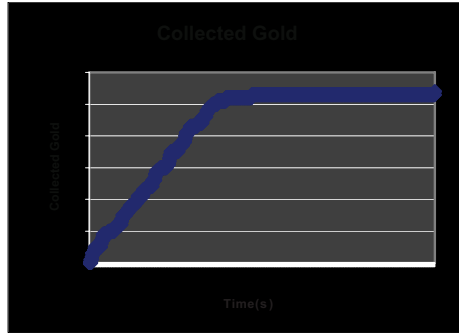
## 5.2 Evaluating Simultaneity of Invocations

Simultaneity of invocation means that an agent may be involved in multiple plan invocations (at the top level) dynamically and simultaneously. In experiment 2, we use a world of 20×20 squares which contain 40 wumpuses and 60 pieces of gold. Rather than randomly generate the world, it has been specifically constructed to contain more difficult scenarios therefore need close cooperative behaviors. These are 25 pieces of unreachable gold in the wumpus world. For example, a piece of gold in the corner is surrounded by two wumpuses. Such gold is unreachable unless a path is opened by killing one of the surrounding wumpus. Again, agents initially have no knowledge of the squares.

We define two teams pursuing different top level goals. Team T1 consists of two agents, ag<sub>1</sub> and ag<sub>2</sub>, who scan a wumpus world and collect the gold found respectively. Team T2 only consists of agent ag<sub>3</sub> who is able to kill wumpuses but has a separate goal that is irrelevant to the main point of this example. We represent it simply by random motion through the world. Three plans, `randommove`, `scancollect` and `kill`, are separate team processes for T1 and T2. Two teams start plans

scancollect and randommove simultaneously (at the top level). With proactive communication from  $ag_1$  [4, 18],  $ag_3$  dynamically finds plan kill for a goal and builds and executes a script to start it many times to help  $ag_2$  while  $ag_3$  is still involved in plan randommove. The team collects 54 pieces of gold finally. Figure 6 shows the distribution of the number of pieces of gold collected over the execution time.

We note that T2 was considered a separate team from T1, rather than as a second sub-team of a larger team, to show the more general situation with separate teams. The entire scheme works within sub-teams of a team as well.



**Fig. 6.** The Distribution of Gold Collected Over the Execution Time

## 6 Conclusions

We have defined role and role variable and accommodated them to complex actions. Based on these conceptual notations, the knowledge of team processes can be specified without having to specific agents and agent variables. This will allow joint mental states to be formed and the teamwork knowledge to be reused by different teams of agents. Based on roles and role variables, we have developed mechanisms of task decomposition and task delegation, by which the knowledge of a team process is decomposed into the knowledge of a team process for individuals and then delegate it to agents. We have developed an efficient representation of joint mental states by which agents only maintain individual processes complementary with others' individual process and a low level of overlapping for team organizations.

## References

- [1] Becht, M.; Muscholl, M; Levi, P, Transformable Multi-Agent Systems: A Specification Language for Cooperation Processes, in Proceedings of the World Automation Congress, ISOMA'98.
- [2] Biddle, B. J. and Thomas, E. J. editors. Role theory: Concepts and research. John Wiley & Sons, 1966.

- [3] Busetta, P., Ronnquist, R., Hodgson, A., Lucas, A., JACK intelligent agents - components for intelligent agents in Java, Technical Report TR9901, Agent Oriented Software, Australia, 1999.
- [4] Cao, S., Volz, R. A., Ioerger, T. R., Miller, M. S., On Proactive Helping Behaviors in Teamwork. IC-AI 2005: 974-980.
- [5] Cohen, P. R. and Levesque, H. J., Teamwork. *Nous*, Special Issue on Cognitive Science and AI, 25(4):487-512, 1991
- [6] Dastani, M., Dignum, V. and Dignum, F. Role Assignment in Open Agent Societies, AAMAS'03.
- [7] Grosz, B. and Kraus, S., Collaborative Plans for Complex Group Actions, *Artificial Intelligence*, 86 (2): 269-357, 1996.
- [8] Kumar, V., Algorithms for Constraint-Satisfaction Problems: A Survey, pp. 32-44, *AI Magazine*, Spring 1992.
- [9] [Pynadath, D. V., Tambe, M., Chauvat, N. and Cavedon, L., Toward team-oriented programming, *Agent Theories, Architectures, and Languages*, 1999, pp. 233-247.
- [10] Russell, S. and Norvig, P., *Artificial Intelligence - A Modern Approach*, Prentice Hall, Upper Saddle River, NJ, 2002.
- [11] Scerri, P., Pynadath, D. V., Schurr, N. and Farinelli, A., Team oriented programming and proxy agents: the next generation, in *Proc. of the 1st Inter. Workshop on Prog. MAS at AAMAS'03*.
- [12] Stone, P. and Veloso, M., Task Decomposition and Dynamic Role Assignment for Real-Time Strategic Teamwork, *Artificial Intelligence*, 100 (2): 2441-273, 1999.
- [13] Tambe, M., Towards Flexible Teamwork, *Journal of artificial intelligence research*, 7: 83-124, 1997.
- [14] Tidhar, G., Team Oriented Programming: Preliminary Report, in Technical Report 41, AAIL, Australia, 1993.
- [15] Tuomela, R., On the Structural Aspects of Collective Action and Change by Logic Programs, *Theory and Decision*, 32 (2): 269-357, 1992.
- [16] Vazquez-Salceda, J., Dignum, V. and Dignum, F., Organizing Multiagent Systems, AAMAS, 11: 307-360, 2005.



# Distributed Collaborative Filtering for Robust Recommendations Against Shilling Attacks

Ae-Ttie Ji<sup>1</sup>, Cheol Yeon<sup>1</sup>, Heung-Nam Kim<sup>1</sup>, and Geun-Sik Jo<sup>2</sup>

<sup>1</sup> Intelligent E-Commerce Systems Laboratory,  
Department of Computer Science & Information Engineering, Inha University  
{aerry13,entireboy,nami}@eslab.inha.ac.kr

<sup>2</sup> School of Computer Science & Engineering, Inha University,  
253 Yonghyun-dong, Incheon, Korea 402-751  
gsjo@inha.ac.kr

**Abstract.** Recommender systems enable a user to decide which information is interesting and valuable in our world of information overload. Collaborative Filtering (CF), one of the most successful technologies in recommender systems suffers from improper use of personal information and the incredibility of recommendations. To deal with these issues, we have been focusing on the trust relationships between individuals, i.e. *web of trust*, especially for protecting the recommender system against profile injection attack. Based on trust propagation scheme, we proposed *TCFMA* architecture which is added agent-based scheme obtaining attack resistance property as well as improving the efficiency of distributed computing. In *web of trust*, users' personal agents find a unique migration path made up of latent neighborhoods and reduce search scope to a reasonable level for mobile agents by using the *Advogato* algorithm. The experimental evaluation on *Epinions.com* datasets shows that the proposed method brings significant advantages in terms of dealing with profile injection attack without any loss of prediction quality.

## 1 Introduction

In a flood of information, a recommender system helps users to decide which items are most valuable and interesting to them. Collaborative Filtering (CF), one of the most successful technologies in recommender systems, has been applied to numerous commercial recommender systems. Even though they are popular, there are problems of improper use of personal information and the incredibility of recommendations especially in centralized CF recommender systems where all ratings by users are owned by system providers [12]. These problems can be partially improved by a distributed personal recommender, but the distributed systems might be vulnerable to a shilling attack, i.e. profile injection attack as similar as centralized ones. That is, an attacker may make many profiles with biased ratings with a malicious intent to influence the recommendations [1, 13, 14]. Because most of recommender systems are open Web services, a malicious attacker can easily inject manipulated profiles [13, 14]. One effective way to protect the system against such an attack is to build *web of trust* among the individuals in order to use only the profiles of trustworthy users [8, 11].

In this paper, we propose *TCFMA (Trust-based Collaborative Filtering with Mobile Agents)* architecture used a distributed CF method in peer-to-peer network using *web of trust* in order to effectively offer the corresponding user trustworthy recommendations. By using the *Advogato* trust metric [4], we propagate the trust information for overcoming the sparseness of *web of trust* as well as obtaining resistance from attacks by the malicious users [3, 4]. In addition, we employ mobile agents to increase the efficiency of distributed computing.

The rest of this paper is organized as follows: The next section contains a brief overview of some related work. In Section 3, we describe our proposed method in detail. Then, the performance evaluation compared with other P2P approaches is presented in Section 4. Finally, Section 5 draws some conclusion of this paper with a discussion of future work.

## 2 Backgrounds and Related Works

**A Robustness Analysis of Collaborative Filtering.** Most of Web-based CF recommender systems employ profiles which are made by anonymous unauthenticated users. That is, the systems can be vulnerable to manipulation due to profiles which are built and injected by an attacker. Many recent researches have shown that commonly used CF algorithms are significantly affected by modest attacks [14]. There are two formal types of profile injection attack that can be defined according to the intent of an attacker: a push attack and a nuke attack [13, 14]. The former makes particular items promoted and the latter makes them demoted in order to be more or less recommended. In the case of centralized CF systems where all profiles are owned by a merchant, a system provider can be a “push” attacker. Because a merchant always wants a customer to buy an item that maximizes profits. *PocketLens* [1], which we benchmark, described that this concern can be met by a distributed personal recommender because only a user can own and controls his or her own profiles. Based on the credibility of recommendations, diverse distributed recommender system architectures and an incremental computing algorithm applicable to those architectures were proposed through this work [1].

**Trust in Recommender Systems.** Even though the distributed recommender can partially improve the effects of profile injection attacks from a system provider, it is still not safe from an anonymous attacker [1]. In order to overcome the vulnerabilities of CF systems to attacks, a number of recent studies focus on the notion of “trust” in recommendation [14]. Calculating explicit trust and reputation values of users or eliciting trust relationships between users, a system employs only the owner’s profile guaranteed identity and trustworthiness [3, 4, 6, 8, 15]. In a more global view, “trust” of a recommender system has been studied in terms of automated attack detection schemes and robustness of recommendation algorithms in the face of malicious attacks [3, 4, 13, 14, 17]. Another perspective about trust focuses on correlation between trust and user similarity. Recent research of Sinha et al. confirms the fact that people tend to prefer recommendations from friends and acquaintances to those from online recommender systems [6]. Moreover, Ziegler et al. shows that people’s preferences can be more similar to preferences of trusted users than those of arbitrary users [7]. That is, using

“trust” for recommender system can be effective in terms of attack-resistance and recommendation quality. However, *web of trust* tend to be sparse; so a mechanism of trust propagation is required [2]. R.Guha et al. define four atomic trust propagation methods that can be applied repeatedly to obtain a final matrix with trust and distrust information [2]. Ziegler et al. protect against massive attacks from malicious users propagating trust effectively in a social network by proposing the *Appleseed* algorithm based on the *Advogato* trust metric [3].

### 3 Trust-Based Collaborative Filtering for P2P Networks

Fig. 1 illustrates a brief overview of our proposed system with three steps; Firstly, a user agent finds the migration path along which a mobile agent migrates in *web of trust*. Along the path, the mobile agent finds trusted neighborhoods. Then the user agent builds a similarity model incrementally for its user by using the information that the mobile agent gathers from neighbors. Finally, the user agent provides its user with recommendations and updates the model with his or her feedback.

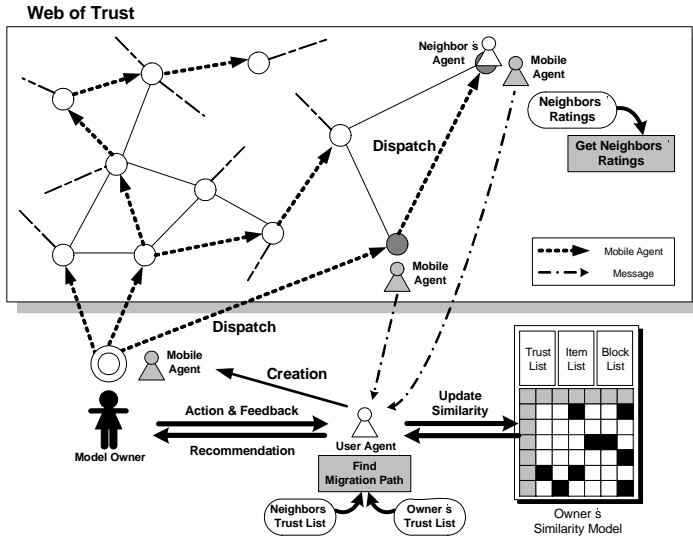


Fig. 1. Overview of trust-based collaborative filtering with mobile agents

#### 3.1 Trust-Based User Selection

In this section we describe a scheme to finding a unique migration path, which consists of the trusted users, for mobile agent of a target user. Before describing the algorithms, some definitions of the notations used herein are introduced.

Let us assume a peer-to-peer network where each user *trusts* other users. The set of  $\{TRUST_{P_X}\}$ , that is the list of users trusted by each user  $P_X$ , simply presented in the

**Table 1.** The meaning of notations

$P_X$	Arbitrary user included in web of trust
$P_O$	Target user, i.e. similarity model owner
$P_C$	Current user who $P_O$ 's mobile agent is visiting at the moment
$\{TRUST_{P_X}\}$	List of users who are trusted by $P_X$
$\{BLOCK_{P_X}\}$	List of users who are distrusted by $P_X$
$\{ITEMS_{P_X}\}$	List of $\langle item, rating \rangle$ pairs, i.e. items which $P_X$ already has expressed his or her own opinion and these preference ratings.
$\{PATH_{P_X}\}$	Migration path which $P_X$ 's mobile agent migrates along
$AGENT_{P_X}$	Personal agent of $P_X$
$AGENT^M_{P_X}$	Mobile agent of $P_X$

same fashion as shown in *Advogato*<sup>1</sup> and *Epinions.com*<sup>2</sup>. *Web of trust* is constructed in the form of a bidirectional graph based on the set of  $\{TRUST_{P_X}\}$ . In contrast with other systems, only a personal agent  $AGENT_{P_X}$  includes  $\langle item, rating \rangle$  pairs, i.e. items in which  $P_X$  is interested and preference ratings for these items, as listed in  $\{ITEMS_{P_X}\}$ .

$AGENT_{P_X}$  finds the migration path  $\{PATH_{P_X}\}$  that includes users trusted by  $P_X$  for a mobile agent  $AGENT^M_{P_X}$ . The *Advogato maximum flow algorithm* is exploited to obtain the migration path for a mobile agent. This algorithm, inspired by the *Ford-Fulkerson maximum flow algorithm*, was used to discover which users are trusted by credible members of an online community and which are not [3, 4, 15]. Because the bidirectional graph of trusts is restructured to form a tree-structure in the process of finding maximum flow through the edges, the algorithm makes it possible to find a unique migration path and to reduce search scope to reasonable levels for mobile agent.

The procedure to find  $\{PATH_{P_X}\}$  is as follows: Assume that  $P_O$ , who is the target user, is the trust source. The capacities  $C$  are assigned to every user in web based on the shortest-path distance from the source to  $P_X$ . The capacity of the source, which can be optionally chosen by  $P_O$ , is based on the number of all users expected to be visited and whose information is used for recommendations. Each successive level has a capacity equal to that of the preceding level  $L$  divided by the average number of trust edges extending from nodes of  $L$ . In order to apply the *Ford-Fulkerson maximum flow algorithm* to this single-source/multiple-target graph with capacity-constrained nodes, it has to be restructured to a single-source/single-target one with capacity-constrained edges rather than nodes. Each node  $P_X$  is split into  $P_X^+$  and  $P_X^-$ , and the capacity that is the original  $C(P_X)$  minus one is assigned to an edge between them. Then, a unit capacity edge is added from  $P_X^-$  to a virtual single target node. The original edge from  $P_X$  to  $P_Y$  is represented as the one from  $P_X^+$  to  $P_Y^-$  with infinite capacity.  $AGENT_{P_O}$  applies the algorithm

<sup>1</sup> *Advogato* <http://www.advogato.org/>

<sup>2</sup> *Epinions.com* <http://www.epinions.com/>

to this converted graph tracing the shortest paths to the target first and adds the nodes reached by network flow to  $\{PATH_{P_O}\}$  [3, 4, 15].

Owing to the *bottleneck property* proposed as a common feature of attack-resistance trust metrics in [15], *Advogato* algorithm is useful to make profile injection attacks take no effects. The *bottleneck property* is that “the total trust quantity accorded to an  $s \rightarrow t$  edge is not significantly affected by changes to the successors of  $t$  [3, 15],” i.e., the number of biased nodes accepted depends only on the number of precedence nodes, not on the number of biased ones. Assume that there is an attacker who intends to promote or demote a particular item, or just to make the overall system function poorly. Even though he builds many profiles (manipulated nodes  $t$ ) with fraud ratings coincided with his purpose, he cannot make  $s$  trust  $t$  manipulated by the attacker. Therefore, the amount of trust accorded to  $t$  dose not increase even though the attacker injects more nodes [3, 4]. For this reason, even the least of profiles that make the attack succeeded is not included in the process of collaboration. More about attack-resistance properties of various trust metrics are discussed in detail in [15] and [17], it has been claimed that *PageRank* [16] possesses *bottleneck property* like *Advogato*.

### 3.2 Incremental Model Building

Recommender algorithm can be characterized by the neighbors they choose for each user, the model they build based on those neighbors, and the way they use the model to form recommendations [1]. In our research, the neighbors of target user  $P_O$  are chosen from the users included in  $\{PATH_{P_O}\}$ .  $P_O$ 's personal agent  $AGENT_{P_O}$  creates a mobile agent,  $AGENT^M_{P_O}$ , to find neighbors and build a similarity model based on them incrementally.  $AGENT^M_{P_O}$  involves only the least amount of information for model building;  $\{PATH_{P_O}\}$  and  $\{ITEMS_{P_O}\}$ . Searching for the users in  $\{PATH_{P_O}\}$  is done by the *Depth First Search* algorithm. Supposing that  $P_C$  is a currently visited neighbor,  $AGENT^M_{P_O}$  traces the path recursively until no users exist in  $\{PATH_{P_O}\} \cap \{TRUST_{P_C}\}$ . Then  $AGENT^M_{P_O}$  is disposed of from the last node after visiting all users in  $\{PATH_{P_O}\}$ .

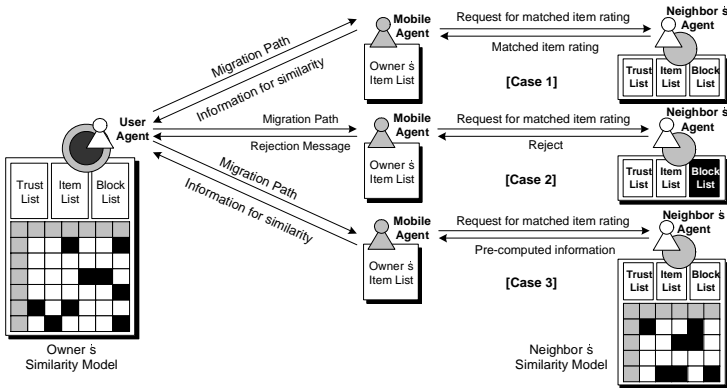


Fig. 2. Agents' tasks in each case

The similarity model is composed of a set of similarities between pairs of items and represented as a matrix [1]. For incremental computation of similarity relationships, each agent does as follows;

1.  $AGENT^M_{P_O}$  identifies  $IO$  and  $IP$  that are  $\{ITEMS_{P_C}\} \cap \{ITEMS_{P_O}\}$  and  $\{ITEMS_{P_C}\} - \{ITEMS_{P_O}\}$  respectively, by communicating with a neighbor's agent  $AGENT_{P_C}$ .
2. For each pair  $(IO_i, IP_j)$ , which is  $IO_i \in IO$  and  $IP_j \in IP$ ,  $AGENT^M_{P_O}$  calculates values shown in Equation 1 and send the values to its own agent  $AGENT_{P_O}$ .

$$\begin{aligned} W_{IO_i, IP_j} &= (Rating_{P_C, IO_i} \times Rating_{P_C, IP_j}) \\ W_{IO_i, IO_i} &= (Rating_{P_C, IO_i})^2 \\ W_{IP_j, IP_j} &= (Rating_{P_C, IP_j})^2 \end{aligned} \quad (1)$$

3.  $AGENT_{P_O}$  adds up these values incrementally until  $AGENT^M_{P_O}$  sends the values of all users in  $\{PATH_{P_O}\}$  except for those which don't have  $IO_i$ .

$$\begin{aligned} W_{Numer} &= W_{Numer} + W_{IO_i, IP_j} \\ W_{Denom1} &= W_{Denom1} + W_{IO_i, IO_i} \\ W_{Denom2} &= W_{Denom2} + W_{IP_j, IP_j} \end{aligned} \quad (2)$$

4.  $AGENT_{P_O}$  calculates the similarity of item pair  $(IO_i, IP_j)$ .

$$sim(IO_i, IP_j) = \frac{\vec{IO_i} \bullet \vec{IP_j}}{|\vec{IO_i}| |\vec{IP_j}|} = \frac{W_{Numer}}{\sqrt{W_{Denom1}} \sqrt{W_{Denom2}}} \quad (3)$$

The above-mentioned procedure involving cosine-similarity metrics can have different versions simply by modifying the second process. For instance, in order to use adjusted cosine similarities between two items as a similarity metric [5], Equ.(1), (2) are modified as

$$\begin{aligned} W_{IO_i, IP_j}' &= (Rating_{P_C, IO_i} - AvgRating_{P_C}) \times (Rating_{P_C, IP_j} - AvgRating_{P_C}) \\ W_{IO_i, IO_i}' &= (Rating_{P_C, IO_i} - AvgRating_{P_C})^2 \\ W_{IP_j, IP_j}' &= (Rating_{P_C, IP_j} - AvgRating_{P_C})^2 \end{aligned} \quad (4)$$

In general cases,  $AGENT_{P_O}$  can update a similarity model owned by  $P_O$  incrementally according to the above procedures. However, assume that a neighbor  $P_C$  has the list  $\{BLOCK_{P_C}\}$ , the list of the users whom  $P_C$  distrusts, and  $P_O$  is included in this list. Not trusting  $P_O$ ,  $P_C$  may not want to be open with  $P_O$  about his own information. In this case,  $AGENT_{P_C}$  rejects the request for information from  $AGENT^M_{P_O}$ .

If the similarity model has already been built for  $P_C$  to get recommendation,  $AGENT^M_{P_O}$  has no need to visit  $P_C$ 's successive nodes and can get their values from this model. This model has been built based on  $\{PATH_{P_C}\}$  setting  $P_C$  as a source.  $\{PATH_{P_C}\}$  is much more likely to include the same users as  $\{PATH_{P_O}\}$ ; the closer the distance from the source to  $P_C$ , the more similar it is. In this case,  $P_O$ 's similarity model might include the information of more neighbors than the users whom  $AGENT^M_{P_O}$  intended to visit at the beginning. Moreover, these users included in  $P_C$ 's

model might overlap with the users who  $AGENT^M_{P_O}$  has already visited or is supposed to visit hereafter. However, “overlapping users” means that they are trusted by more preceding level users, which can have a positive influence on recommendations by reflecting their opinions more. By pruning the successive nodes, tracing costs can be decreased drastically.

### 3.3 Propagating User Feedback

Based on this similarity model,  $AGENT_{P_O}$  provides recommendations to  $P_O$ . Simply, the particular items, which obtain either the highest averages of each column or the highest prediction values, are recommended. Explicit prediction values of user  $P_O$  for item  $IP_j$  can be computed by the weighed sum of  $P_O$ 's ratings about  $IO_i$  using the similarity  $sim(IO_i, IP_j)$  as the weight and defined as Equ.(5) [1, 5].

$$p\_rating_{P_O, IP_j} = \frac{\sum_{IO_i \in IO} \{sim(IO_i, IP_j) \times Rating_{P_O, IO_i}\}}{\sum_{IO_i \in IO} |sim(IO_i, IP_j)|} \quad (5)$$

The idea is that the average rating of items that are similar to the selected item is a good estimate of the rating for the selected item [1].

One of the principle issues that we have to consider in a model-based approach is how the updated information can be reflected in original models during the term when they have not been re-built yet. When item  $IP_k$  is recommended to  $P_O$ , she can express her preference for this item as a rating. Whenever the  $P_O$  gives feedback,  $AGENT_{P_O}$  deletes  $IP_k$ 's column from her model and adds it to  $\{ITEMS_{P_O}\}$  because it now has its own rating. Updated information that is,  $\{ITEMS_{P_O}\}$  including a pair  $\langle IP_k, P_O$ 's rating about  $IP_k \rangle$ , is propagated to personal agents of users in  $\{TRUST_{P_O}\}$ . From  $P_C$ 's point of view who is given this information by  $AGENT_{P_O}$ , item  $IP_k$  has not been included in  $\{ITEMS_{P_O}\}$  before, so the similarities between  $IP_k$  and items in  $\{ITEMS_{P_O}\} \cap \{ITEMS_{P_C}\}$  cannot be included in  $P_C$ 's model.

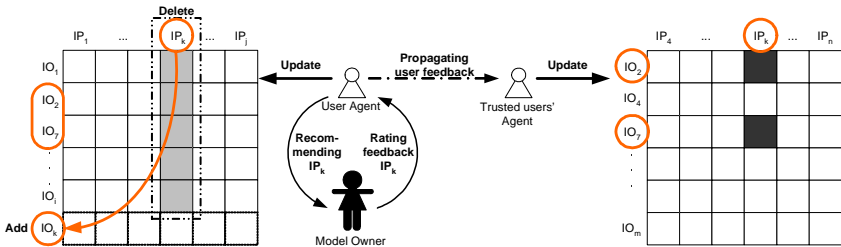


Fig. 3. Recommendations and propagation user's feedback

$AGENT_{P_C}$  now can compute similarities between them, update the model in patches; incremental updating can be achieved by propagating it to the users in  $P_C$ 's own list  $\{TRUST_{P_C}\}$ . Fig.3 illustrates feedback and update process.

## 4 Experimental Results

In this section we present the results of applying *TCFMA* method for recommendation in a peer-to-peer environment. The prototype system is implemented using *IBM Aglet* Software with *JDK1.4.2* [10].

### 4.1 Data Sets and Evaluation Metrics

*Epinions.com* is an online community where users can review various items and rate them on a scale of 1 to 5. Judging whether the reviews of others are helpful to users themselves or not, users can express trust or distrust of these reviewers. We collected the dataset by crawling the *Epinions.com* site in May 2006. The collected dataset was too sparse to be used for experiments, so we selected a dataset including users who had rated at least 5 items and expressed a trust opinion of at least 25 users. In addition, the items had been rated by at least 10 users, i.e. the dataset contained 121,862 ratings for 2,955 items and 216,490 trust information presented by 4,751 users. The sparsity level of our dataset is  $1 - (121,862 / (4751 \times 2955))$ , which is 0.9913. Then, this dataset was divided into two parts; training set contains all of each user's ratings except one rating used for testing. Testing set contains the only one rating for each user.

In order to measure performance, *Mean Absolute Error*, which is used as a measure of how accurate prediction of user's rating for an item can be, was performed [1, 5, 8]. *MAE* of all users in the testing set is defined as:

$$MAE = \frac{\sum_{i=1}^M |p\_rating_i - a\_rating_i|}{M} \quad (6)$$

where  $M$  is a list of all items and  $\langle a\_rating_i, p\_rating_i \rangle$  is the actual/predicted rating pairs of each user in the testing set.

Another metric *Absolute Prediction Shift* measured the distortion of prediction occurring due to an attack. While  $p\_rating$  is the predicted rating computed before an attack,  $p\_rating'$  means the predicted rating computed after an attack [14].

$$APS = \frac{\sum_{i=1}^M |p\_rating_i - p\_rating'_i|}{M} \quad (7)$$

The evaluation value of *Prediction Shift* originally has two meanings, in other words, a positive value has different meaning from a negative value. Each value means that the attack has succeeded in making the target item more positively or negatively rated [14]. However, *APS* measures the absolute value just to evaluate the influence from injected profiles regardless of attack-classification in our experiments.

### 4.2 Preliminary Experiments

**Overall Performance of Prediction Quality.** Increasing the number of users used for similarity model building, we compared our system with one of the methods proposed in *PocketLens*. The experiment was carried out in order to indicate that the proposed method performs as good as an existing work. Prior to experiments, the vector of ratings  $\vec{r}$  for each user was normalized as  $\|\vec{r}\| = 1$  except for experiments by using



an adjusted cosine-based similarity metric; otherwise, users who had rated a large number of items had more influence than users who had only rated a few items [9]. In the process of model building, mobile agents find neighbor peers based on the converted trust graph and compute similarity relations by using a cosine-based similarity metric and an adjusted cosine-based one (see Equation 1 and 4). On the other hand, in the benchmarked one, the information of the peers randomly accessed regardless of *web of trust* is used for model building by using the cosine-based one.

**Table 2.** Overall performance of prediction quality

Neighbor peer size	10	30	50	70	100
Random	1.2866	1.2863	1.2859	1.2859	1.2859
TCFMA + cosine	1.2113	1.2114	1.2100	1.2101	1.2101
TCFMA + adjusted	1.2384	1.2480	1.2412	1.2415	1.2402

As shown in Table 2, all three methods provided nearly the same values for MAE. When the *TCFMA + cosine-based* scheme was used, the results moved from 1.2113 to 1.2101, which shown better prediction quality than other two methods. It can be observed that the proposed methods provide more accurate predictions than random model building at all neighborhood size levels. For example, when the neighborhood size is 50, the *TCFMA + cosine-based* scheme and the *TCFMA + adjusted cosine-based* one obtain an MAE of 1.2100 and an MAE of 1.2412 whereas the random scheme obtains an MAE of 1.2859.

In addition, the results show that even a small number of users can build relatively better model with our proposed methods, whereas the random scheme needs more users' information to obtain a similarity model of stable.

**Positive Effect of Trust for Prediction.** When the dataset including the users who have many trust opinions is used for building a similarity model, the model includes a larger number of trustworthy users. In order to determine the sensitivity of trust opinion size on the quality of the prediction, we assumed that each user have trust users of only the number of  $x$ . In each step of evaluations, the trust opinion size was selectively varied for building a similarity model of each testing user. According to the value of  $x$ , the prediction quality in cases of the proposed methods, i.e. similarity model building based on the converted trust graph, was evaluated.

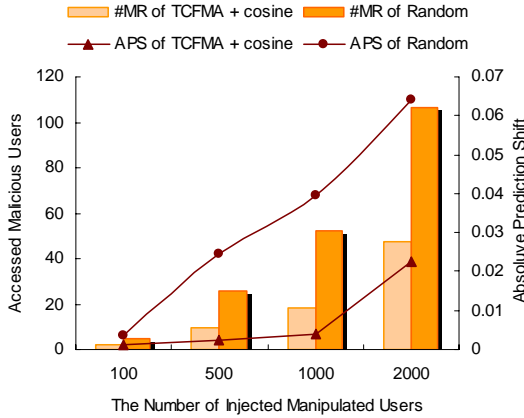
**Table 3.** Sensitivity of trust on MAE (neighbor peer size = 50)

Trust $x$	Trust 5	Trust 10	Trust 15	Trust 25	Trust 45
TCFMA + cosine	1.4131	1.3338	1.3313	1.2867	1.1611
TCFMA + adjusted	1.5688	1.3028	1.2952	1.2512	1.2238

The conclusion drawn from these results shown in Table 3 is that the more trust opinions are included in each user, the better prediction quality is obtained. This means that the *direct* trust opinions have positive influence on the prediction quality.

### 4.3 Performance Evaluations

**Robustness against Profile Injection Attack.** To evaluate the robustness against a malicious attack, a set of manipulated user profiles including arbitrary 50 ratings each was inserted into the original training dataset, while the number of these profiles was increased from 100 to 2000. Each user in the manipulated set was made to have the trust edges to all users in the set while some users in the original set were made to present trust to some of manipulated users. Through the prior experiment and previous research, we selected 50 as a neighborhood size and set it up in this experiment [1, 9]. In both cases of *TCFMA + cosine-based* and random model building, the average number of manipulated users accessed for building each user’s model was measured and the distortion of prediction occurring due to these data was compared by using *Absolute Prediction Shift*.



**Fig. 4.** Comparison of robustness on manipulated users

In accordance with increases in the number of manipulated users, we find that the proposed method showed better results than random model building in both measurements. Fig. 4 illustrates that the proposed method significantly outperforms others in resistance against the profile injection attack. For example, when the number of manipulated users was 2000, *TCFMA + cosine-based* scheme obtained a *#MR* of 47.71 and an *APS* of 0.0224, whereas the random scheme obtained a *#MR* of 106.56 and an *APS* of 0.0639.

**Efficiency of similarity model building.** Finally, we focused on the time required for model building. Forwarded along the path, a mobile agent only sends computed results to the personal agent of the model owner by the proposed methods whereas in the random scheme request/response messages have to be exchanged between the model owner and each random user for similarity model building. The experiment was conducted to evaluate the time and the number of accessed users that are required

**Table 4.** Comparison of required time and accessed users (neighbor user size = 50)

Model Owner		User 1	User 2	User 3	User 4	User 5	Average
TCFMA + cosine	Time(ms)	5786.81	11576.54	9776.97	12676.54	9425.59	9848.49
	# User	292.64	861.94	680.08	953.54	636.64	684.968
Random	Time(ms)	31590.24	30129.18	31966.27	23209.48	20977.24	27574.48
	# User	4379.48	4209.75	4505.89	3315.13	2962.29	3874.51

to build similarity models. For a  $\langle item, rating \rangle$  pair of each 5 users in a testing set, the similarity model was built 30 times relatively to get the average performance.

The experimental results show that the average time of *TCFMA + cosine-based* model building is 9848.49 (msec.), which is reasonably shorter than 27574.48 (msec.) of the random one. In addition, the number of users who have to be accessed for similarity model building is also considerably smaller as shown in Table 4. These results demonstrate that the proposed method is far superior with respect to the effectiveness of similarity model building.

## 5 Conclusion and Future Work

In a peer to peer environment, a distributed recommender system is an ongoing area of diverse applications [1]. In the paper, we propose a novel *TCFMA* architecture to solve the problems that can occur in online collaborative filtering recommender systems related to an improper use of personal information and a profile injection attack. In order to obtain more trustworthy and accurate recommendations, we consider the trust relationships between users in *web of trust*. The *Advogato* trust metric is used as a trust propagation scheme required for overcoming sparseness of trust information. As noted in our experimental results, we obtain extraordinary robustness from malicious attacks without any degradation of prediction quality, compared to general peer-to-peer collaborative filtering recommender system. Moreover, we also achieve an efficiency of distributed computing for building item-item similarity model by employing trust-based collaborative filtering scheme which is added some useful functionality of mobile agents.

However, there still remains certain issue: trust decay. It means that the trust relationship becomes weaker as it forwards to its successors [2, 3]. In our system, although a mobile agent finds trust neighbors who are in the closest-distance from a model owner first, the neighbors, even on the different levels, are regarded as the same. It is essential to take this phenomenon into consideration for applying trust propagation algorithm to real-world application. Another interesting issue is about attack detection, which has been often shown in recent studies. Automated attack detection algorithms based on diverse types of attack model can lead more robust recommendation algorithms [13, 14, 17]. As the recommender systems have been more common in E-commerce application, these issues are becoming more interesting and important.

## References

1. Miller, B., Konstan, J., Terveen, L., Riedl, J.: PocketLens: Towards a Personal Recommender System. In *ACM Transactions on Information Systems* 22 (2004) 437-476
2. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of Trust and Distrust. In *Proc. of the 13<sup>th</sup> Int. Conf. on World Wide Web* (2004), ACM Press
3. Ziegler, C-N., Lausen, G.: Propagation Models for Trust and Distrust in Social Networks. In *Information Systems Frontiers Vol. 7*, Springer Netherlands (2005) 337 – 358
4. Levien, R., Aiken, A.: Attack Resistant, Scalable Name Service. Draft submission to the 4<sup>th</sup> Int. Conf. on Financial Cryptography (2000)
5. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based Collaborative Filtering Recommendation Algorithms. In *Proc. of the 10<sup>th</sup> Int. Conf. on World Wide Web* (2001)
6. Sinha, R., Swearingen, K.: Comparing Recommendations Made by Online Systems and Friends. In *Proc. of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries* (2001)
7. Ziegler, C-N., Lausen, G.: Analyzing Correlation between Trust and User Similarity in Online Communities. In *Proc. of the 2<sup>nd</sup> Int. Conf. on Trust Management, LNCS, Vol. 2995* (2004) 251-265
8. Massa, P., Avesani, P.: Trust-aware Collaborative Filtering for Recommender Systems. In *Proc. of Int. Conf. on Cooperative Information Systems* (2004)
9. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering. In *Proc. of the 22<sup>nd</sup> ACM SIGIR Conf. on Research and Development in Information Retrieval* (1999)
10. Lange, D.B., Oshima, M.: *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley (1998)
11. Jung, J. J.: Visualizing recommendation flow on social networks. *Journal of Universal Computer Science*, Vol. 11, No. 11 (2005) 1780-1791
12. Kim, H.J., Jung, J.J., Jo, G.S.: Conceptual Framework for Recommendation System based on Distributed User Ratings. *LNCS, Vol. 3032* (2003) 115-122
13. O'Mahony, M., Hurley, N., Kushmerick, N., Silvestre, G.: Collaborative Recommendation: A Robustness Analysis. *ACM Transactions on Internet Technology*, Vol. 4, No. 4 (2004) 344-377
14. Mobasher, B., Bruke, R., Bhaumik, R., Williams, C.: Towards Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness. to appear in *ACM Transactions on Internet Technology*, Vol. 7, No. 2 (2007)
15. Leivien, R.: Attack Resistant Trust Metrics. Ph.D thesis, UC Berkeley, Berkeley, CA, USA (2003)
16. Page, L., Brin, S., Motwani, R., Winograd, T.: The Pagerank Citation Ranking: Bringing Order to the Web. Technical Report, Stanford Digital Library Technologies Project (1998)
17. Twigg, A., Dimmock, N.: Attack-resistance of computational trust models. In *Proc. of the 12<sup>th</sup> IEEE Int. Workshop on Enabling Technologies* (2003) 275-280

# Competition and Coordination in Stochastic Games

Andriy Burkov, Abdeslam Boularias, and Brahim Chaib-draa

DAMAS Group  
Dept of Computer Science  
Laval University  
G1K 7P4, Quebec, Canada  
{burkov,boularia,chaib}@damas.ift.ulaval.ca

**Abstract.** Agent competition and coordination are two classical and most important tasks in multiagent systems. In recent years, there was a number of learning algorithms proposed to resolve such type of problems. Among them, there is an important class of algorithms, called adaptive learning algorithms, that were shown to be able to converge in self-play to a solution in a wide variety of the repeated matrix games. Although certain algorithms of this class, such as Infinitesimal Gradient Ascent (IGA), Policy Hill-Climbing (PHC) and Adaptive Play  $Q$ -learning (APQ), have been catholically studied in the recent literature, a question of how these algorithms perform versus each other in general form stochastic games is remaining little-studied. In this work we are trying to answer this question. To do that, we analyse these algorithms in detail and give a comparative analysis of their behavior on a set of competition and coordination stochastic games. Also, we introduce a new multiagent learning algorithm, called ModIGA. This is an extension of the IGA algorithm, which is able to estimate the strategy of its opponents in the cases when they do not explicitly play mixed strategies (e.g., APQ) and which can be applied to the games with more than two actions.

## 1 Introduction

Competition and coordination between autonomous agents are two classical and most important tasks in multiagent systems. Coordination is especially important in multi-robotic systems where a number of non-adversarial robots (but not necessarily explicitly cooperative) are aimed to accomplish a task while being limited in communication and in knowledge about principles of rationality underlying their counterparts. On the other hand, competition is a natural condition of most real life situations. The agents that share limited resources, negotiate about prices and, in general, have proper interests first or last find themselves in a competitive situation.

Typically, multiagent environments are modeled as *stochastic games* [1]. Stochastic game is a model to represent multi-state multiagent environments having Markovian property and a stochastic inter-state transition rule, and can

be used to model inter-agent interactions in such environments. Formally, a stochastic game is a tuple  $(n, \mathbf{S}, \mathcal{A}^{1 \dots n}, T, R^{1 \dots n})$ , where  $n$  is the number of agents,  $\mathbf{S}$  is the set of states  $\mathbf{s} \in \mathbf{S}$  now represented as vectors,  $\mathcal{A}^j$  is the set of actions  $a^j \in \mathcal{A}^j$  available to agent  $j$ ,  $\mathbf{A}$  is the joint action space  $\mathcal{A}^1 \times \dots \times \mathcal{A}^n$ ,  $T$  is the transition function:  $\mathbf{S} \times \mathbf{A} \times \mathbf{S} \mapsto [0, 1]$ ,  $R^j$  is the reward function for agent  $j$ :  $\mathbf{S} \times \mathbf{A} \mapsto \mathbb{R}$  and  $\mathbf{s}_0 \in \mathbf{S}$  is the initial state.

Further, in the article we will refer to a game theoretic terminology, therefore let's introduce some useful notions of the Game Theory here. A matrix game is a tuple  $(n, \mathcal{A}^{1 \dots n}, R^{1 \dots n})$ , where  $n$  is the number of players,  $\mathcal{A}^j$  is the strategy space of player  $j$ ,  $j = 1 \dots n$ , and the value function  $R^j : \mathcal{A}^1 \times \dots \times \mathcal{A}^n \mapsto \mathbb{R}$  defines the utility for player  $j$  of a joint action  $\mathbf{a} \in \mathbf{A} = \mathcal{A}^1 \times \dots \times \mathcal{A}^n$ . A *mixed* strategy for player  $j$  is a distribution  $\pi^j$ , where  $\pi_{a^j}^j$  is the probability for player  $j$  to select some action  $a^j$ . A strategy is *pure* if  $\pi_{a^j}^j = 1$  for some  $a^j$ . A *strategy profile* is a collection  $\mathbf{\Pi} = \{\pi^j | j = 1 \dots n\}$  of all players' strategies. A *reduced profile for player  $j$* ,  $\mathbf{\Pi}^{-j} = \mathbf{\Pi} \setminus \{\pi^j\}$ , is a strategy profile containing strategies of all players except  $j$ , and  $\mathbf{\Pi}_{\mathbf{a}^{-j}}^{-j}$  is the probability for players  $k \neq j$  to play a joint action  $\mathbf{a}^{-j} \in \mathbf{A}^{-j} = \mathcal{A}^1 \times \dots \times \mathcal{A}^{j-1} \times \mathcal{A}^{j+1} \times \dots \times \mathcal{A}^n$  where  $\mathbf{a}^{-j}$  is  $\langle a^k | k = 1 \dots n, k \neq j \rangle$ .

Recently, there was a number of learning algorithms proposed to resolve decision problems in stochastic games [1,2,3,4,5,6,7,8,9,10,11]. Typically, these algorithms are constructed to iteratively *play* a game with an opponent, and, by playing this game, to *converge* to a solution. Solution in game theory is called *equilibrium*. We say that the playing strategies of all agents forms an equilibrium in a stochastic game if a unilateral deviation of an agent from its current strategy contradicts its principles of rationality (usually, maximization of the utility).

Among the learning algorithms proposed for the stochastic games, there is an important class, which we call *adaptive learning algorithms*, that are proven to be able to converge in self-play (i.e., when learning “against” agents that are using the same learning algorithm) to an equilibrium solution in a wide variety of repeated matrix games. The advantage of the adaptive learning algorithms with respect to other class of multiagent learning algorithms, such as *equilibrium learning algorithms* [1,7,8], is that the latter are calculating an equilibrium solution regardless the other agents' actual behavior (i.e., equilibrium learners assume that their opponents are rational, though they may not be) and their convergence is limited to a number of cases where these equilibria are identifiable. The adaptive learning agents, on the contrary, make no assumptions about their opponents' rationality and learning capabilities, and about the solution type they are searching. Adaptive agents are adapting to their opponents and a solution is found as an *emerging result* of this adaptation. Among adaptive algorithms, the most outstanding and theoretically sound ones are Infinitesimal Gradient Ascent (IGA) [4], Policy Hill-Climbing (PHC) [2] and Adaptive Play  $Q$ -learning (APQ) [3]. These algorithms were empirically tested by their respective authors on the different test benches. However, although these algorithms were tested on a number of repeated matrix games and on some examples of stochastic games, a number of questions is remaining. First, whether these

algorithms are well extensible to the general form stochastic games. Second, how these algorithms are comparable between themselves (in terms of convergence and relative effectiveness against each other).

In this paper we are trying to answer these questions. To do that, we analyse these algorithms in detail and give a comparative analysis of their behavior on a set of competition and coordination stochastic games, which includes two-robot-on-the-grid coordination game and two-robot-predator-prey competition game. Further, we introduce a new multiagent learning algorithm, called ModIGA, a modification of the IGA algorithm.

## 2 Adaptive Learning Algorithms

As we noted above, to learn a “good” policy in stochastic games a number of adaptive algorithms have been proposed. They can be conventionally divided into three groups: (1) Opponent Modelling algorithms [3,5], (2) Policy Gradient based algorithms [2,4] and Adaptivity Modelling algorithms [9,10,11]. Although the algorithms of the third group are very interesting and empirically shown to have several attractive properties, such as exploiting their opponents in adversarial games [10,11] and converging to a solution maximizing welfare of both players in non-adversarial two-player matrix games [11], there are still no theoretical proofs of their correctness, while in the first two groups there are algorithms that were formally proven to have such properties as rationality and convergence. In our analysis, we opted for the following three adaptive learning algorithms: Infinitesimal Gradient Ascent (IGA) [4], Policy Hill-Climbing (PHC) [2] and Adaptive Play  $Q$ -learning (APQ) [3] because, as we have just noted, (1) they are theoretically proven to converge to a stable solution (at least in self-play), (2) they represent two major classes of learning algorithms, those able to play pure strategies only (APQ) and those able to play mixed strategies (IGA, PHC).

In this section we analyze in detail these algorithms. Also, we introduce a new multiagent learning algorithm, called ModIGA. This is an extension of the IGA algorithm, which is able to estimate the strategy of its opponents in the cases when they do not explicitly play mixed strategies (e.g., APQ) and which, unlike IGA, can be applied to the games with more than two actions.

### 2.1 Adaptive Play $Q$ -Learning

Formally, each player  $j$  playing Adaptive Play [12] saves in memory a history  $H_t^j = \{\mathbf{a}_{t-p}^{-j}, \dots, \mathbf{a}_t^{-j}\}$  of the last  $p$  joint actions played by the other players. To select a strategy to play at time  $t+1$  each player randomly and irrevocably samples from  $H_t^j$  a set of examples of length  $l$ ,  $\hat{H}_t^j = \{\mathbf{a}_{k_1}^{-j}, \dots, \mathbf{a}_{k_l}^{-j}\}$ , and calculates the empiric distribution  $\hat{\Pi}^{-j}$  as an approximation of the real reduced profile of strategies played by the other players, using the following:

$$\hat{\Pi}_{\mathbf{a}^{-j}}^{-j} = \frac{C(\mathbf{a}^{-j}, \hat{H}_t^j)}{l} \quad (1)$$

where  $C(\mathbf{a}^{-j}, \hat{H}_t^j)$  is the number of times that the joint action  $\mathbf{a}^{-j}$  was played by the other players according to the set  $\hat{H}_t^j$ . Given the probability distribution over the other players' actions,  $\hat{\Pi}^{-j}$ , the player  $j$  plays its best reply,  $BR^j(\hat{\Pi}^{-j})$ , to this distribution with some exploration. If there are several equivalent best replies, the player  $j$  randomly chooses one of them. Young [12] proved the convergence of Adaptive Play to an equilibrium when played in self-play for a big class of games such as the coordination and common interest games.

Adaptive Play  $Q$ -learning (APQ) is an extension of Young's algorithm to the multi-state stochastic game context. To do that, the usual single-agent  $Q$ -learning update rule [13] was modified to consider multiple agents as follows:

$$Q^j(s, \mathbf{a}) \leftarrow (1 - \alpha)Q^j(s, \mathbf{a}) + \alpha[R^j(s, \mathbf{a}) + \gamma \max_{a^j \in \pi^j(s')} U^j(\hat{\Pi}(s') \cup \{\pi^j(s')\})]$$

where  $j$  is an agent,  $\mathbf{a}$  is a joint action played by the agents in state  $s \in \mathcal{S}$ ,  $Q^j(s, \mathbf{a})$  is the current value for player  $j$  of playing the joint action  $\mathbf{a}$  in state  $s$ ,  $R^j(s, \mathbf{a})$  is the immediate reward the player  $j$  receives if the joint action  $\mathbf{a}$  is played in the state  $s$  and  $\pi^j(s')$  are all possible *pure* strategies that are available for player  $j$  in state  $s'$ .

## 2.2 Infinitesimal Gradient Ascent

To examine the dynamics of using policy gradient in repeated games, Singh, Kearns and Mansour modeled this process for two-player, two-action matrix games. They called their approach Infinitesimal Gradient Ascent (IGA) [4]. Unlike APQ, which can learn and play pure strategies only, IGA players were designed to be capable to learn and play mixed strategies.

The problem of the gradient ascent in matrix games was modelled by Singh and colleagues as having two payoff matrices for the row and column players,  $r$  and  $c$ , as follows:

$$R^r = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}, \quad R^c = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

If row player  $r$  selects an action  $i$  and the column player  $c$  selects an action  $j$ , then the payoffs they obtain are  $R_{ij}^r$  and  $R_{ij}^c$  respectively.

Because the game being modelled has only two available actions for each agent, a mixed strategy can be represented as a single value. If we let  $\alpha \in [0, 1]$  be a probability the player  $r$  selects the action 1, then  $1 - \alpha$  will be the probability to play the action 2. Similarly, we can define as  $\beta \in [0, 1]$  and  $1 - \beta$  the probabilities to play actions 1 and 2 by the player  $c$ . The expected utility of playing a strategy profile  $\{\alpha, \beta\}$  for player  $r$  can then be calculated as follows:

$$U^r(\{\alpha, \beta\}) = r_{11}\alpha\beta + r_{22}(1 - \alpha)(1 - \beta) + r_{12}\alpha(1 - \beta) + r_{21}(1 - \alpha)\beta$$

At each game iteration, to estimate the effect of changing its current strategy, player  $r$  calculates a partial derivative of the expected utility with respect its current mixed strategy:

$$\frac{\partial U^r(\{\alpha, \beta\})}{\partial \alpha} = \beta u - (r_{22} - r_{12})$$



where  $u = (r_{11} + r_{22}) - (r_{21} + r_{12})$ .

Having calculated the gradient, IGA agent adjusts its current strategy in the direction of this gradient as to maximize its utility:

$$\alpha_{t+1} = \alpha_t + \eta \frac{\partial U^r(\{\alpha_t, \beta_t\})}{\partial \alpha}$$

where  $\eta$  is a step size, usually  $0 < \eta \ll 1$ . Similar equations can be written for the column player  $c$  as well. Obviously, the opponent's mixed strategy is supposed to be known by the players.

Singh and colleagues proved the convergence of IGA to an equilibrium (or, at least, to the equivalent average reward of an equilibrium), when played in self-play, in the case of the infinitesimal step size ( $\lim_{\eta \rightarrow 0}$ ).

### 2.3 Policy Hill-Climbing Algorithm

The first practical algorithm capable to play mixed strategies that realized the convergence properties of IGA was Policy Hill-Climbing (PHC) learning algorithm [2]. The PHC algorithm requires neither knowledge of the opponent's current stochastic policy nor its recently executed actions (the latter is required for the APQ algorithm, for example). The algorithm, in essence, performs hill-climbing in the space of mixed strategies and is, in fact, a simple modification of the single-agent  $Q$ -learning technique. It is composed of two parts. The first part is the reinforcement learning component, which is based on the  $Q$ -learning technique to maintain the values of the particular actions in the states:

$$\hat{Q}^j(\mathbf{s}_t, a_t^j) \leftarrow (1 - \alpha) \hat{Q}^j(\mathbf{s}_t, a_t^j) + \alpha \left[ R_t^j(\mathbf{s}_t, a_t^j) + \gamma \max_{a_{t+1}^j} \hat{Q}(\mathbf{s}_{t+1}, a_{t+1}^j) \right]$$

The second part is the game theoretic component, which maintains the current mixed strategy in each system's state. The policy is improved by increasing the probability that the agent selects the highest valued action, by using the small step  $\delta$  which is called learning rate:

$$\pi_{a^j}^j(\mathbf{s}) \leftarrow \pi_{a^j}^j(\mathbf{s}) + \Delta_{\mathbf{s}a^j} \quad (2)$$

where

$$\Delta_{\mathbf{s}a^j} = \begin{cases} -\delta_{\mathbf{s}a^j} & \text{if } a^j \neq \operatorname{argmax}_{a'^j} \hat{Q}(\mathbf{s}, a'^j) \\ \sum_{a'^j \neq a^j} \delta_{\mathbf{s}a'^j} & \text{otherwise} \end{cases} \quad (3)$$

$$\delta_{\mathbf{s}a^j} = \min \left( \pi^j(\mathbf{s}, a^j), \frac{\delta}{|\mathcal{A}^j| - 1} \right) \quad (4)$$

while constrained to a legal probability distribution. If  $\delta = 1$  the algorithm is equivalent to the single-agent  $Q$ -learning as soon as the learning agent will deterministically execute the best action (greedy policy). As well as the single-agent  $Q$ -learning, this technique is rational and converges to the optimal solution if the other players follow a fixed (stationary) policy. However, if the other players are learning, the PHC algorithm may not converge to a stationary policy though its average reward will converge to the reward of a Nash equilibrium [2].

## 2.4 ModIGA

While IGA demonstrated good convergence results, its applicability in reality is limited to the two-action case where the opponent is playing an *identifiable mixed strategy*. This assumption does not reflect real nature problems. In reality, we are usually expecting agent to observe the opponent's actions rather than its mixed strategy. Furthermore, the real life learning agents, as well as their counterparts, are intended to have more than two available actions.

We introduce an improved version of the IGA algorithm, which is able to learn a mixed strategy for more than two simple actions and to estimate the strategy of its opponents even if they do not explicitly play a mixed strategy. (This is the case, for example, when playing against APQ algorithm.)

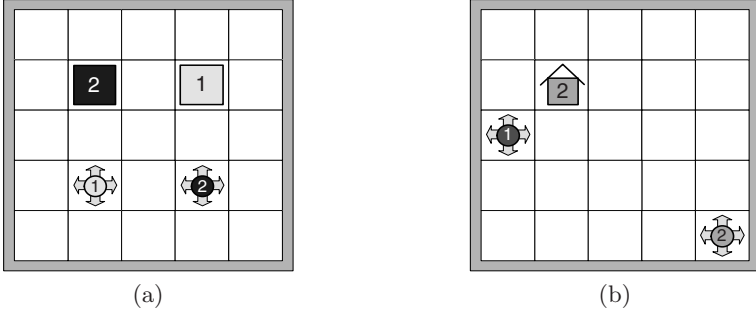
To make IGA agent able to estimate the strategy of its opponent we used the Adaptive Play's probability estimation technique described in Subsection 2.1. Having calculated the estimation of the opponent's strategy,  $\Pi_{t+1}^j$ , the IGA agent is able to calculate the gradient of its own current strategy by using the equations of Subsection 2.2. It is important to note that even if the opponent is not playing explicitly mixed strategies (e.g., APQ), the IGA agent using this technique is still able to calculate the gradient of its strategy, though this gradient will be calculated to the averaged opponent's strategy rather than to its real strategy.

When there are more than two actions at the agents' disposal, the techniques of gradient calculation and strategy update of two-action case do not work well and, as we observed, cannot be readily extended to the case of multiple actions. First, this is because in this case there can not be one variable to represent the agent's strategy and another one depending on it. Second, in the two-action case, an increase of the probability to make one action tacitly and at the same degree decreased the probability of the other action to be executed, which always kept the total probability equal to 1. In the multiple action case, this is no longer so.

To deal with this problem, we adapted the technique used in PHC algorithm. It consists in updating the strategy in the direction of the action with the higher  $Q$ -value (see equations 2,3,4). But unlike PHC, in our ModIGA algorithm,  $\delta$  is proportional to the  $Q$ -value. This keeps the gradient ascent property, i.e., the step in the direction of the gradient is proportional to the gradient itself.

## 3 Examples

To make our experiments, we programmed two stochastic games, which model two the most important types of multiagent interactions: coordination and competition. The first game is called two-robot-on-the-grid coordination problem, first introduced by Hu and Wellman [7]. The game consists of the *grid* containing a number of *cells*. There are *two robots* on the grid, which have four available actions, *up*, *down*, *left* and *right*. By making actions, robots are able to transit between cells with a certain probability of the transition success. If transition is successful, robot changes the cell in the intended direction. Otherwise, robot keeps its current position. For each action made in each cell, except the goal cell, robot



**Fig. 1.** (a) The two-robot-on-the-grid coordination problem and (b) The two-robot-predator-prey competition game

receives a *negative reward*. A *collision* is possible if robots are trying to transit into the same cell or to trade cells. In the case of collision, robots receive a negative *collision reward*. The goal of each robot hence is to reach its respective goal cell by collecting the minimal value of negative reward. In our experiments we set the following values of the parameters of the model. The action reward in all non-goal cells is  $-0.04$  and is  $0$  in the goal cell, the collision reward is  $0.1$ , the probability of action success is  $0.9$  and the discount factor is  $0.95$ . The configuration of the grid and the start and goal cells of robots are depicted in Figure 1(a).

The second stochastic game we programmed is called two-robot-predator-prey competition game. In this game, there are two robots on the same grid as in the coordination game, but the robots play different roles. The first robot (player 1) is called “predator” and its goal in the game is to catch (to achieve a collision with) the second robot, called “prey”. The goal of the “prey” (player 2) is to reach a refuge where it cannot be caught. I.e., the *goal situations* for both robots are opposite. If the predator has achieved its goal (i.e., caught the prey) its reward for any action in this state is  $0$  and the prey, in turn, receives a negative reward of  $-1$  for any action. On the other hand, if the prey has reached the refuge, its reward for any action in this state is  $0$  and the reward of the predator is  $-1$  regardless its position and action. In all other states robots receive a negative reward of  $-0.04$  for any action. So, we see, that this game is strictly competitive. We set the following values of the other parameters of the model. The probability of action success of predator was set to  $0.9$ , the same parameter of the prey was set to  $0.65$ . These values equalize the chances of winning of both predator and prey, as it was determined in self-play (when both predator and prey used the same learning algorithms). The discount factor was set to  $0.95$ . The configuration of the grid, the start cells and the refuge cell for the prey are depicted in Figure 1(b).

## 4 Experiments

In our experiments we compared the convergence processes and the final solution quality of all algorithm pairs (i.e., IGA versus IGA, IGA versus PHC, and so on)

in the both environments presented in Figure 1. The curves in Figures 2-5 show the results of these experiments.

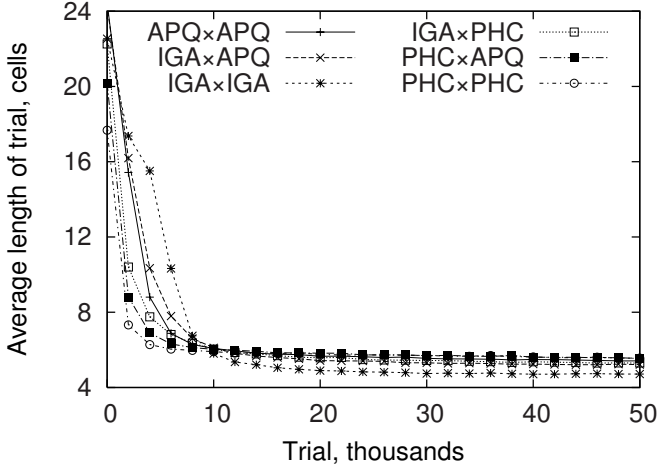
Figures 2-3 show the results of the experiments in the two-robot-on-the-grid coordination problem. The curves represent the average number of inter-cell transitions of player 1 in one trial as a function of the number of trials. To build each curve, we averaged data over 10 similar experiments. (The results are shown for the first agent only, because the curves for the second agent are the same.) To reflect the convergence speed of each algorithm pair, Figure 2 represents the first 50,000 trials of the learning process. We can easily see that the IGA×IGA pair converges slower than the other pairs, and, on the contrary, the pair PHC×PHC demonstrates the fastest convergence speed. This can be explained by the fact that the learning space of the APQ and IGA algorithms is  $|\mathbf{S}||\mathcal{A}|^2$ , since they learn in the space of joint actions, instead of  $|\mathbf{S}||\mathcal{A}|$  of the PHC algorithm, which considers its own actions only.

Figure 3 reflects the final 100,000 trials of the same learning processes. These curves reflect the solution quality of each algorithm pair. We can see here that whereas PHC demonstrated the faster convergence speed in the first learning trials, all algorithm pairs with a participation of PHC demonstrated a worse final solution quality, i.e., in these curves, the final value of average trial length is higher than this for the algorithm pairs without PHC. On the other hand, the cases APQ×APQ and IGA×IGA demonstrated the best average solutions. This can be explained by the fact that both APQ and IGA can observe the actions of their opponents, and, by so doing, to adapt better to the strategy of the opponent. Moreover, since in the two-robot-on-the-grid problem the solution is deterministic (a pair of trajectories) and APQ learns pure strategies directly, it is obvious that in that case the solution found by APQ×APQ cannot be worse than the others.

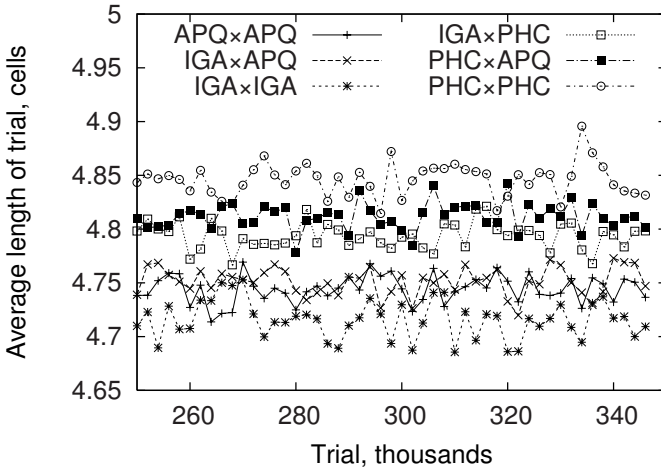
In our opinion, the results obtained, in particular the empirical convergence of the algorithms of different types against each other, are very interesting, because there have been no theoretical guarantees that these algorithms converge when playing not against themselves. This could be explained by the similarity of the convergence curves of these algorithms in self-play. Hence, the policies generated at the end of each trial differ not much. Thus, the agents had almost the same behavior when we combined these different algorithms in one play. Additionally, the convergence properties can be held in this situation, because the agents were not able to distinguish whether the other agent was using the same algorithm or not.

Figures 4-5 show the results of the experiments in the two-robot-predator-prey problem. As in the coordination problem's case, we tested all the possible two-by-two combinations of the chosen algorithms. The curves represent average trial length of the predator agent. For the same reasons as stated above, we did not present the curves for the prey agent.

Similarly to the results obtained in the coordination game, in this adversarial game we observed the convergence to a stable value for each algorithm pair. Because of the same factors, the convergence speed during the first 50,000 trials was

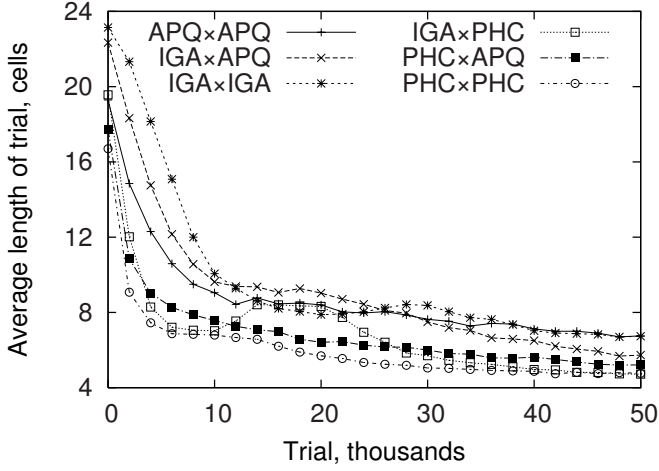


**Fig. 2.** The optimal trajectory learning in a  $5 \times 5$  two-robot-on-the-grid game. The curves reflect the length of a trial as a function of the trials number, where the agents use the algorithms PHC, IGA/ModIGA and APQ: the first 50,000 trials.

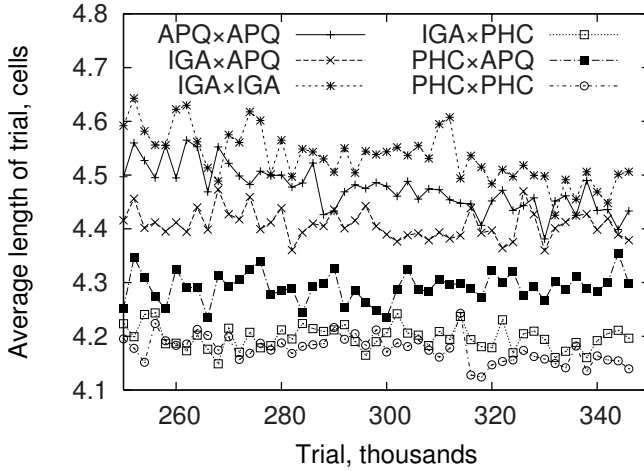


**Fig. 3.** The optimal trajectory learning in a  $5 \times 5$  two-robot-on-the-grid game. The curves reflect the length of a trial as a function of the trials number, where the agents use the algorithms PHC, IGA/ModIGA and APQ: the final 100,000 trials.

the slowest for the IGAxIGA algorithm pair and the fastest for the PHCxPHC case (Figure 4). However, in terms of the solution quality (final value of average trial length), the results are inverse. All the algorithm pairs with a participation of PHC (PHCxPHC, PHCxAPQ and IGAxPHC) behaved better than those without PHC during the last 100,000 learning trials (Figure 5). We explain this by the ability of PHC to learn mixed strategies, which can bring better solutions



**Fig. 4.** The dynamics of learning in the two-robot-predator-prey game, with a  $5 \times 5$  grid. The curves show the length of a trial as a function of the trials number, where the agents use the algorithms PHC, IGA/ModIGA and APQ.



**Fig. 5.** The dynamics of learning of the last 100,000 trials in the two-robot-predator-prey game, with a  $5 \times 5$  grid. The curves show the length of a trial as a function of the trials number, where the agents use the algorithms PHC, IGA/ModIGA and APQ.

in adversarial games than pure strategies can do. For the same reasons, APQ cannot perform better than PHC in this case. But, surprisingly for us, IGA $\times$ IGA case demonstrated the longest average trial length at the end of the learning, which is somewhat unexpected, since its convergence properties are the same as for the PHC algorithm. This fact is remaining for further investigation.

**Table 1.** Effective running time in different games, in seconds.

Game	PHC×PHC	PHC×IGA	PHC×APQ	IGA×IGA	IGA×APQ	APQ×APQ
Coordination	69	98	113	128	117	153
Adversarial	86	121	147	171	280	218

Finally, we measured the average running time of all experiments (Table 1). As expected, the PHC algorithm was the fastest in terms of calculation time (it is, in fact, the simplest in terms of the amount of calculations required at each iteration). APQ was, as expected, the slowest among all algorithms in both competition and coordination games, since at each iteration it performs a computationally hard operation of the opponent strategy estimation.

## 5 Conclusion and Future Work

In this work we compared different multiagent learning algorithms in play in two different stochastic games, a coordination game and an adversarial game. To do that, we extended Infinitesimal Gradient Ascent algorithm [4] to the case where the environment has multiple states and the agents can execute more than two different actions. The other two algorithms, namely Policy Hill Climbing [2] and Adaptive Play  $Q$ -learning [3] have already been adapted to the stochastic game setting by their respective authors. These algorithms was proven to converge to an equilibrium in self-play in the repeated matrix games, but, to our knowledge, they were never compared with each other in the case of stochastic games. This encouraged us to do this research. The goals we aimed were to investigate these algorithms in detail and to make a preliminary conclusion about their performance in stochastic games when playing against each other.

The first important observation, which we noted as a result of our experiments, is that these algorithms converge in play against each other, which was not observed and theoretically proved before. The second observation is the different quality of the solutions found by the different algorithm pairs.

In terms of execution time, we observed that the PHC algorithm required less time to get a decision in each state and, thus, it converged more quickly in the examples we used in this work. On the other hand, in the cooperation game the algorithms, which were able to observe the actions of their opponents (i.e., AQP and IGA) learned better solutions in terms of the average trajectory length, than PHC which had not such ability.

In our future work we would like to focus our attention to the finding of the formal convergence properties of these algorithms when used one against other. Also, we would extend our experiments to the more complex and unpredictable environments and to the algorithms using the learning principles other than adaptivity to the opponent’s current policy, such as Hyper- $Q$  [10] and some non-stationary algorithms such as [14,15].

## References

1. Littman, M.: Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the Eleventh International Conference on Machine Learning (ICML'94), New Brunswick, NJ, Morgan Kaufmann (1994)
2. Bowling, M., Veloso, M.: Multiagent learning using a variable learning rate. *Artificial Intelligence* **136**(2) (2002) 215–250
3. Gies, O., Chaib-draa, B.: Apprentissage de la coordination multiagent : une méthode basée sur le Q-learning par jeu adaptatif. *Revue d'Intelligence Artificielle* **20**(2-3) (2006) 385–412
4. Singh, S., Kearns, M., Mansour, Y.: Nash convergence of gradient dynamics in general-sum games. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI'94), San Francisco, CA, Morgan Kaufman (1994)
5. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98), Menlo Park, CA, AAAI Press (1998)
6. Hu, J., Wellman, P.: Multiagent reinforcement learning: Theoretical framework and an algorithm. In: Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98), San Francisco, CA, Morgan Kaufmann (1998)
7. Hu, J., Wellman, M.: Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research* **4** (2003) 1039–1069
8. Littman, M.: Friend-or-foe Q-learning in general-sum games. In: Proceedings of the Eighteenth International Conference on Machine Learning (ICML'01), San Francisco, CA (2001) Morgan Kaufman
9. Chang, Y., Kaelbling, L.: Playing is believing: The role of beliefs in multi-agent learning. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS'01), Canada (2001)
10. Tesauero, G.: Extending Q-learning to general adaptive multi-agent systems. In: Thrun, S., Saul, L., Scholkopf, B., eds.: *Advances in Neural Information Processing Systems*. Volume 16., Cambridge, MA, MIT Press (2004)
11. Burkov, A., Chaib-draa, B.: Effective learning in adaptive dynamic systems. In: Proceedings of the AAAI 2007 Spring Symposium on Decision Theoretic and Game Theoretic Agents (GTDT'07), Stanford, California (2007) To appear.
12. Young, H.: The evolution of conventions. *Econometrica* **61**(1) (1993) 57–84
13. Watkins, C., Dayan, P.: Q-learning. *Machine Learning* **8**(3) (1992) 279–292
14. Powers, R., Shoham, Y.: New criteria and a new algorithm for learning in multi-agent systems. In: Saul, L.K., Weiss, Y., Bottou, L., eds.: *Advances in Neural Information Processing Systems*. Volume 17., MIT Press (2005)
15. Powers, R., Shoham, Y.: Learning against opponents with bounded memory. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05). (2005)



# Multiagent-Based Dynamic Deployment Planning in RTLS-Enabled Automotive Shipment Yard

Jindae Kim<sup>1</sup>, Changsoo Ok<sup>1</sup>, Soundar R.T. Kumara<sup>1</sup>, and Shang-Tae Yee<sup>2</sup>

<sup>1</sup> Department of Industrial and Manufacturing Engineering, The Pennsylvania State University  
Univeristy Park, PA 16802, USA

{jxk447, cuo108, skumara}@psu.edu

<sup>2</sup> Manufacturing Systems Research Laboratory, General Motors  
Warren, MI 48090, USA

shang-tae.yee@gm.com

**Abstract.** Real-time vehicle location information enables to facilitate more efficient decision-making in dynamic automotive shipment yard environment. This paper proposes a multiagent-based decentralized decision-making model for the vehicle deployment planning in a shipment yard. A multiagent architecture is designed to facilitate decentralized algorithms and coordinate different agents dynamically. The results of computational experiments show that the proposed deployment model outperforms a current deployment practice with respect to the deployment performance measures.

## 1 Introduction

Recently, radio frequency identification (RFID) technologies have been widely introduced into real world supply chains [7][9]. RFID-enabled real time locating system (RTLS) provides a real-time visibility of dynamics by tracking all the entities of supply chains and allowing instant identification and automatic information transfer of the entities' state [1]. Providing the visibility of dynamics and the state of supply chain entities in an automated and timely manner gives new opportunities to better control the dynamic supply chain [4][5][8].

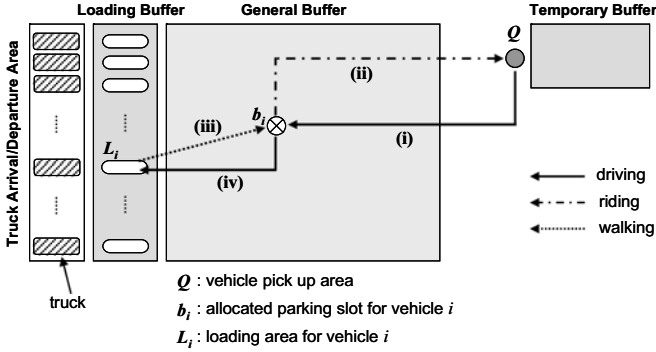
This paper introduces a deployment planning in an automotive shipment yard as a practical application. A finished vehicle is deployed in a particular slot of the shipment yard of a plant until it is shipped to a destination. The shipment yard is equipped with RTLS, where an active RF tag attached to a vehicle transmits RF signals to the readers and then, the RF processor determines the vehicle's information including its location. RTLS can facilitate the exchange of necessary information in real-time. Due to the real-time availability, a new decision-making approach needs to be adaptable and flexible to make decisions in constantly changing operational environment of the shipment yard. The objective of this paper is to develop a new decision-making model for improving finished vehicle deployment in the RTLS-enabled shipment yard with the real-time information from RTLS. A proposed multiagent computational architecture facilitates decentralized algorithms and coordinates vehicle agents dynamically using a market-based model.

## 2 Vehicle Deployment Planning

Once a vehicle is produced from an assembly plant, the vehicle is moved to a temporary buffer in a shipment yard and then, deployed to a general buffer. The general buffer stores a number of deployed vehicles until they are moved to loading buffers where they are loaded onto trucks for delivery. A vehicle is manufactured with its delivery destination and priority. The loading area and the loading schedule of the vehicle are determined by its delivery destination and priority. To reduce order-to-delivery lead-time, a loading schedule is determined based on two general dispatching rules, high-priority-first-loaded (HPFL) rule and first-come-first-loaded (FCFL) rule.

### 2.1 Description of Deployment Operations

For a single vehicle, the deployment process is divided into four elementary operations conducted by yard workers. As illustrated in Figure 1, they are: (i) driving vehicle  $i$  from pick up area  $Q$  in a temporary buffer to allocated parking slot  $b_i$  in a general buffer; (ii) riding back to the temporary buffer from  $b_i$  by a transportation utility like a van; (iii) walking from determined loading area  $L_i$  to slot  $b_i$ ; and (iv) driving vehicle  $i$  from  $b_i$  to its loading area  $L_i$ . The operations, (i) and (ii), are conducted by yard workers belonging to a temporary buffer, namely group A, and (iii) and (iv) are performed by workers belonging to a loading buffer, namely group B.



**Fig. 1.** Illustration of the elementary operations related to deployment of vehicle  $i$

Based on the understanding of deployment operations in a shipment yard, the deployment decision and the deployment planning are defined as follows;

**Definition 1 (Deployment decision).** A deployment decision is the problem of allocating the available parking slots in a general buffer to a newly produced vehicle to minimize the total operational labor cost to conduct the deployment of vehicle.  $\square$

**Definition 2 (Deployment planning).** A planning for the periodic deployment decisions through the planning time horizon is termed as a deployment planning, where each deployment decision is made interactively. Thus, the objective of deployment planning is to achieve better long term performance by controlling deployment decisions in dynamic environments.  $\square$

## 2.2 Performance Measure of Vehicle Deployment

In this study, a consolidated operational labor cost is used as a performance measure for a vehicle deployment. The consolidated operational labor cost is heavily depends on driving, riding, and walking distance which define the distance between every pair of locations. Driving and riding distances can be approximately calculated as a rectilinear distance because a general buffer is filled with vehicles, and vehicles and vans can move only through the accessible roads in the yard. For example, driving and riding distances between two locations  $\alpha$  and  $\beta$ ,  $d_D(\alpha, \beta)$  and  $d_R(\alpha, \beta)$ , are calculated by  $d_D(\alpha, \beta) = d_R(\alpha, \beta) = |X^\alpha - X^\beta| + |Y^\alpha - Y^\beta|$  where  $X^i$  and  $Y^i$  represent  $X$ -coordinate and  $Y$ -coordinate of location  $i$ , respectively. Walking distance, represented as Euclidian distance, between two locations  $\alpha$  and  $\beta$  is obtained by:  $d_W(\alpha, \beta) = \sqrt{(X^\alpha - X^\beta)^2 + (Y^\alpha - Y^\beta)^2}$ . Using the approximated distance matrices, the consolidated operational labor cost  $C(i)$  for vehicle  $i$  is calculated by:

$$C(i) = \left\{ c_A \cdot \left[ \frac{d_D(Q, b_i)}{s_D} + \frac{d_R(b_i, Q)}{s_R} \right] + c_B \cdot \left[ \frac{d_W(L_i, b_i)}{s_W} + \frac{d_D(b_i, L_i)}{s_D} \right] \right\} \quad (1)$$

where  $s_D$ ,  $s_R$ , and  $s_W$  are driving, riding, and walking speeds (unit distance / unit time), respectively.  $c_A$  and  $c_B$  are unit time labor cost for group A and group B.

## 2.3 Dynamics in Vehicle Deployment Planning

Since vehicles are produced sequentially over time, a series of the decisions for corresponding parking slot allocations are formulated as a dynamic deployment planning problem that consists of a set of deployment decisions. In the dynamic deployment planning problem, deployment decisions made in the current time period affect decisions that will be made later. The dynamic deployment planning problem is mathematically formulated as follows:

$$\text{Minimize } \sum_{i \in T} \sum_{b \in B^i} \left\{ c_A \cdot \left[ \frac{d_D(Q, b)}{s_D} + \frac{d_R(b, Q)}{s_R} \right] + c_B \cdot \left[ \frac{d_W(L_i, b)}{s_W} + \frac{d_D(b, L_i)}{s_D} \right] \right\} \cdot x_b^i \quad (2)$$

$$\text{subject to } \sum_{b \in B^t} x_b^t = 1, \quad \forall t$$

where  $T$  is the set of decision epochs throughout the time horizon for deployment planning, and decision epoch  $t$  implies the time when a newly produced vehicle is released to the temporary buffer.  $B^t$  represents the set of available parking slots at time  $t$ , and a binary decision variable  $x_b^t$  is 1 if a vehicle produced at time  $t$  is assigned to parking slot  $b$  ( $b \in B^t$ ), otherwise 0. In this formulation, the dynamics of the set of available parking slots is represented as  $B^{t+1} = [B^t \setminus \{b_t\}] \cup \hat{B}^{t+1}$ ,  $\forall t$ , where  $b_t$  is the parking slot that is allocated to the vehicle produced at time  $t$ , and  $\hat{B}^{t+1}$  is the set of parking slots that are newly emptied between time  $t$  and  $t+1$ .

Newly emptied parking slots occur due to two main reasons. Parking slots become empty as vehicles deployed in these slots move into loading areas to be shipped out based on loading schedules. In general, once a truck arrives at a shipment yard to load

---

```

FIND  $V^k = \{j \mid L_j = k, j \in V\}$ , where  $V$  is the set of all vehicles in a general buffer
FOR  $m = 1$  To  $T_{capa}$  ( $T_{capa}$  is the maximum capacity of truck)
    FIND  $J^* = \{j^* \mid j^* = \arg \max_{j \in V^k} (p_j)\}$ , where  $p_j$  is delivery priority of vehicle  $j$ .
    IF  $|J^*| = 1$  THEN
        Load the vehicle  $j^*$  in  $J^*$ 
         $V^k = V^k \setminus \{j^*\}$ 
    ELSE IF  $|J^*| > 1$  THEN
        FIND  $j_{FC}^* = \arg \min_{j^* \in J^*} (r_{j^*})$ , where  $r_j$  is production time of vehicle  $j$ .
        Load the vehicle  $j_{FC}^*$ 
         $V^k = V^k \setminus \{j_{FC}^*\}$ 
    END IF
     $m = m + 1$ 
END FOR

```

---

**Fig. 2.** HPFL and FCFL based vehicle loading schedule in a shipment yard

vehicles at loading area  $k$ , a vehicle loading schedule is generated by high-priority-first-loaded (HPFL) rule and first-come-first-loaded (FCFL) rule as described in Figure 2. However, the loading schedule cannot be perfectly predicted because of various uncertainties in vehicle loading environment.

The other reason is that vehicles can be put on hold or returned to the plant because of unexpected product quality problems. Once a quality problem for already deployed vehicles is reported, yard workers hold the related vehicles or move those vehicles to the plant to take corrective actions. Due to above two main reasons, the set of available parking slots keeps dynamically changing throughout the time horizon, and these inherent dynamics in the deployment planning environment lead to consider an adaptable and flexible decision-making model that can handle the dynamics.

## 2.4 Deployment Planning in a Current Shipment Yard

In a current shipment yard, the information about the available parking slots is obtained by manual reporting. Yard workers report the parking slots to a yard manager after they moved vehicles into a general buffer for deployment or into loading areas for shipment. By gathering manually reported information, the yard manager updates the status of the general buffer. Since manual reporting and updating processes take some time, the status of the general buffer is not updated in a real time manner, but updated every certain time period, typically once a day.

In the beginning of each time period, the set of available parking slots  $B$  obtained by manual reporting is provided for the deployment of vehicles that will be produced during that time period. Since vehicle production schedule cannot be predicted with accuracy, a deployment decision is made whenever a finished vehicle is released to a shipment yard. Each decision in this period is made only based on the set of available parking slots  $B$  obtained at the beginning of this period. In other words, even though parking slots may become empty during this period, this information is not available for the deployment decisions made in this period. Now, by solving the following

problem, one of the available parking slots is assigned to vehicle  $i$  so as to minimize a consolidated operational labor cost:

$$\begin{aligned} & \text{Minimize } \sum_{b \in B^i} \left\{ c_A \cdot \left[ \frac{d_D(Q, b)}{s_D} + \frac{d_R(b, Q)}{s_R} \right] + c_B \cdot \left[ \frac{d_W(L_i, j)}{s_W} + \frac{d_D(j, L_i)}{s_D} \right] \right\} \cdot x_b^i \\ & \text{subject to } \sum_{b \in B^i} x_b^i = 1 \text{ and } B^i = B \setminus \tilde{B}^i \end{aligned} \quad (3)$$

where decision variable  $x_b^i$  is 1 if vehicle  $i$  is allocated to slot  $b$  ( $b \in B^i$ ), otherwise 0. In this formulation,  $B^i$ , which denotes the set of available slots for vehicle  $i$ , is obtained from  $B$  by excluding  $\tilde{B}^i$  ( $\tilde{B}^i \subset B$ ), where  $\tilde{B}^i$  is the set of parking slots allocated to the vehicles that are produced earlier than vehicle  $i$  during the period.

### 3 Multiagent-Based Deployment Planning with RTLS

In a new shipment yard, RTLS enables the yard manager to be aware of the information of the vehicles in the yard and the available parking slots in real-time. To handle the dynamics in a vehicle deployment and the real-time information from RTLS, a multiagent-based deployment planning model is proposed, where a market-based mechanism is introduced to improve the initially made deployment decision.

#### 3.1 Initial Deployment Decision

Once a vehicle is newly produced and moved into a temporary buffer, the yard manager obtains the vehicle information, including its delivery priority and loading area and then, allocates the best parking slot among the currently available slots which are automatically updated in a real-time manner by RTLS. An initial deployment decision for a newly produced vehicle  $i$  can be determined by:

$$\begin{aligned} & \text{Minimize } \sum_{b \in B^i} \left\{ c_A \cdot \left[ \frac{d_D(Q, b)}{s_D} + \frac{d_R(b, Q)}{s_R} \right] + c_B \cdot \left[ \frac{d_W(L_i, b)}{s_W} + \frac{d_D(b, L_i)}{s_D} \right] \right\} \cdot x_b^i \\ & \text{subject to } \sum_{b \in B^i} x_b^i = 1 \text{ and } B^i = B_E^i \setminus B_C^i \end{aligned} \quad (4)$$

where decision variable  $x_b^i$  is 1 if parking slot  $b$  ( $b \in B^i$ ) is allocated to vehicle  $i$ , otherwise 0. In the above formulation,  $B^i$ , the set of available slots for vehicle  $i$ , is provided in real-time by detecting the set  $B_E^i$  and  $B_C^i$ , where  $B_E^i$  is the set of all empty parking slots when vehicle  $i$  is produced and  $B_C^i$  denotes the set of parking slots that are empty but allocated to other vehicles currently in the yard, however the vehicles are not completely moved into the slots yet.

#### 3.2 Improvement of Initial Deployment Decision

Once a vehicle is moved into the temporary buffer, it requires some time period until the vehicle is completely moved to a parking slot in the general buffer. This time period is explained by two main operational delays. One is a vehicle needs to stay in a

temporary buffer until a yard worker becomes available to move, and the other is, it takes some operational time to move a vehicle from the temporary buffer to an allocated parking slot. During this time period, the set of available parking slots keeps changing due to the dynamics in shipment yard operations. To improve the adaptiveness and competitiveness of a decision-making model in this dynamic environment, initial decisions made in Section 3.1 are required to be updated in response to those changes.

Since RTLS can capture dynamic events in the yard, whenever a newly emptied parking slot is detected, current parking slot allocations for the vehicles that are not completely moved into a general buffer can be updated by solving:

$$\begin{aligned}
 & \text{Minimize } \sum_{i \in V^p} \sum_{b \in B^p \cup \{b_{new}\}} \left\{ c_A \cdot \left[ \frac{d_D(Q, b)}{s_D} + \frac{d_R(b, Q)}{s_R} \right] + c_B \cdot \left[ \frac{d_W(L_i, b)}{s_W} + \frac{d_D(b, L_i)}{s_D} \right] \right\} \cdot x_b^i \\
 & \text{subject to } \sum_{i \in V^p} x_b^i \leq 1, \quad \forall b, \quad b \in B^p \cup \{b_{new}\} \\
 & \sum_{b \in B^p \cup \{b_{new}\}} x_b^i = 1, \quad \forall i, \quad i \in V^p
 \end{aligned} \tag{5}$$

where decision variable  $x_b^i$  is 1 if parking slot  $b$  is assigned to vehicle  $i$ , otherwise 0, and  $b_{new}$  is the newly detected empty parking slot. In this formulation,  $V^p$  represents the set of vehicles that are produced but not completely moved to a general buffer yet, and  $B^p$  denotes the set of parking slots that are currently allocated to vehicles in  $V^p$ .

The above formulation is based on a centralized information processing and decision-making in which the yard manager should be aware of all the related shipment yard information, such as vehicles with their locations, general buffer availability, and loading areas. Moreover, the above decision problem is required to be solved repeatedly whenever a new empty slot is detected and to provide a solution in time. With increasing dynamics and growing uncertainties in the shipment yard, the centralized approach is inadequate in processing all the distributed information and is unable to make prompt responses to real shipment yard situations.

Therefore, this study proposes a multiagent-based decision-making architecture where a market-based mechanism is facilitated to accommodate the dynamics and to process a large amount of distributed information. A multiagent-based decision-making framework is well known as it follows a nature of decentralization and has been suggested to overcome limitations of the centralized approach [6][10].

### 3.3 Iterative Auction Mechanism to Update Initial Deployment Plan

Market-based control mechanism is a paradigm for controlling distributed and dynamic system by taking advantage of desirable features of a market, including decentralization, interacting agents, and notion of resource that need to be allocated [3]. Due to the nature of the updating process of parking slot allocations, as discussed in Section 3.2, it is very natural to model this process as a negotiation process between competitive vehicle agents who need to acquire parking slots to achieve their individual goals. An iterative auction algorithm proposed by Bertsekas to solve  $n$  to  $n$  allocation problem [2] is introduced as a market-based mechanism in the proposed multiagent-based approach. Since the problem in Section 3.2, is  $n$  vehicle agents to

$n+1$  parking slots allocation problem, we add a dummy vehicle agent to re-design the problem as one-to-one matching allocation problem.

It is supposed that parking slot  $b$  has a price  $p(b)$  and the vehicle agent who takes the slot must pay the price  $p(b)$ . Net value of parking slot  $b$  for vehicle agent  $i$  is represented as a difference between the utility value and the price, that is,  $u_i(b) - p(b)$ . Through the algorithm, parking slots are allocated to vehicle agents so as to maximize the total summation of utility values.

In this model, once a new empty parking slot  $b_{new}$  is detected, the deployment planner agent calls all the vehicle agents in  $V^P$  to request the information of the currently allocated parking slots to the vehicle agents in  $V^P$ . Now vehicle agent  $i$ ,  $i \in V^P$  responds to the deployment planner agent by sending a currently allocated parking slot  $b_i$ . After collecting the set of currently allocated parking slots  $B^P$ , the deployment planner agent opens an auction market by broadcasting the set of all available slots including  $b_{new}$ ,  $B^P \cup \{b_{new}\}$ , to vehicle agents in  $V^P$ .

The auction algorithm proceeds in iterations starting with current allocation where  $b_{new}$  is initially allocated to dummy vehicle  $v_d$  and initial prices of parking slots are set to zero. Each iteration starts with the allocation result and the set of prices taken from previous iteration, and the iteration continues until all vehicle agents are satisfied with an allocation. By introducing  $\varepsilon$ -complementary slackness, vehicle agent  $i$  would be satisfied if the net value of allocated parking slot  $b_i$  is within  $\varepsilon$  of maximum net value:

$$u_i(b_i) - p(b_i) \geq \max_{b \in B^P \cup \{b_{new}\}} \{u_i(b) - p(b)\} - \varepsilon \quad (6)$$

However, it is assumed that dummy vehicle agent  $v_d$  is always satisfied with the any allocation result through the iteration.

In each iteration, if there is any vehicle agent  $i$  not satisfied with the previous allocation result, this vehicle agent finds a slot  $b_i^*$  that provides maximal net value;

$$b_i^* = \underset{b \in B^P \cup \{b_{new}\}}{\text{Arg Max}} \{u_i(b) - p(b)\} \quad (7)$$

Once vehicle agent  $i$  finds the parking slot  $b_i^*$ , she/he exchanges parking slots with the vehicle agent allocated to  $b_i^*$  at the beginning of the iteration by bidding a new price for parking slot  $b_i^*$ . The new price of parking slot  $b_i^*$ ,  $p_{new}(b_i^*)$ , is set to the level where vehicle agent  $i$  is indifferent between  $b_i^*$  and her/his second best parking slot:

$$p_{new}(b_i^*) = p(b_i^*) + (\alpha_i - \beta_i + \varepsilon) \quad (8)$$

where  $\alpha_i$  denotes the net value of the best parking slot for vehicle  $i$  and  $\beta_i$  represents the net value of the second best parking slot. The net values,  $\alpha_i$  and  $\beta_i$ , are calculated as  $\alpha_i = \max_{b \in B^P \cup \{b_{new}\}} \{u_i(b) - p(b)\}$  and  $\beta_i = \max_{b \in [B^P \cup \{b_{new}\}] \setminus \{b_i^*\}} \{u_i(b) - p(b)\}$ , respectively. As shown

in Figure 3, the above process is repeated in a sequence of iterations until all vehicle agents are satisfied with the result of allocations. The strength of an  $\varepsilon$ -complementary slackness iterative auction algorithm is explained by its computational efficiency and optimality properties [2].

---

**STEP 1 : Deployment Planner Agent (Opening a market)**  
 Ask vehicle agent  $i$ , for  $\forall i \in V^p$ , about currently allocated slot  $b_i$   
 Broadcast  $B^p \cup b_{new}$  to vehicle agents in  $V^p$   
 GO TO STEP 2

**STEP 2 : Vehicle Agents (Bidding and Exchanging)**  
 FOR Vehicle agent  $i$ ,  $i \in V^p$   
 IF  $u_i(b_i) - p(b_i) < \max_{b \in B^p \cup \{b_{new}\}} \{u_i(b) - p(b)\} - \varepsilon$  THEN  
     Bid new price  $p_{new}(b_i^*)$  for  $b_i^*$   
     Exchange slots with vehicle agent assigned to  $b_i^*$   
     Set  $p(b_i) = p_{new}(b_i^*)$   
 ELSE  
     Do nothing  
 END IF  
 END FOR  
 GO TO STEP 3

**STEP 3 : Deployment Planner Agent (Terminating a market)**  
 IF  $u_i(b_i) - p(b_i) \geq \max_{b \in B^p \cup \{b_{new}\}} \{u_i(b) - p(b)\} - \varepsilon$ , for  $\forall i \in V^p$  THEN  
     Close a market  
 ELSE  
     GO TO STEP 2  
 END IF

---

**Fig. 3.** A high level description for implementing iterative auction algorithm

### 3.4 Design of Two Different Utility Functions for Vehicle Agent

The utility value of a parking slot for a vehicle agent plays key role in the proposed market-based algorithm. To calculate  $u_i(b)$ , the value of utility when vehicle agent  $i$  takes parking slot  $b$ , two different utility functions are proposed. One is the utility function without consideration of delivery priority,  $u_i^{w/oDP}(b)$ , and the other is the utility function with consideration of delivery priority,  $u_i^{wDP}(b)$ . As defined in Equation (9), the reciprocal of a consolidated operational labor cost is given as the utility function  $u_i^{w/oDP}(b)$ .

$$u_i^{w/oDP}(b) = \left\{ c_A \cdot \left[ \frac{d_D(Q, b)}{s_D} + \frac{d_R(b, Q)}{s_R} \right] + c_B \cdot \left[ \frac{d_W(L_i, b)}{s_W} + \frac{d_D(b, L_i)}{s_D} \right] \right\}^{-1} \quad (9)$$

As described in Section 2.3, vehicles in the general buffer are moved into loading area based on high-priority-first-loaded (HPFL) rule. The HPFL rule based loading schedules cause vehicles to be shipped out by the order of delivery priority. As a result, the amount of time a vehicle with higher delivery priority stays in a general buffer before it is moved into loading area is expected to be shorter than the amount of time a vehicle with lower delivery priority stays, and a vehicle agent with higher delivery priority should have higher utility value for the parking slot than a vehicle agent with lower delivery priority has. With the utility function with consideration of



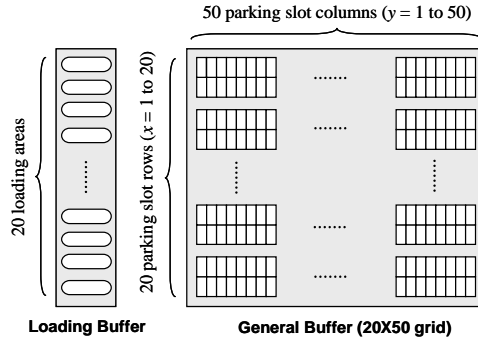
delivery priority, a parking slot that is preferred by multiple vehicle agents can be highly utilized by decreasing the amount of time which a certain vehicle agent stays at this preferred parking slot. This mechanism makes it possible to reduce the overall operational labor cost caused during the deployment planning period. The utility function  $u_i^{wDP}(b)$  is represented as:

$$u_i^{wDP}(b) = (1 + p_i) \cdot \left\{ c_A \cdot \left[ \frac{d_D(Q, b)}{s_D} + \frac{d_R(b, Q)}{s_R} \right] + c_B \cdot \left[ \frac{d_W(L_i, b)}{s_W} + \frac{d_D(b, L_i)}{s_D} \right] \right\}^{-1} \quad (10)$$

where  $p_i$  denotes delivery priority of vehicle agent  $i$ . The value of  $p_i$  is within 0 to 1,  $0 < p_i < 1$ , and larger  $p_i$  represents the higher delivery priority.

## 4 Numerical Experiments

To validate the proposed multiagent-based deployment planning model in the RTLS-enabled shipment yard, computational experiments are conducted using simulations. As illustrated in Figure 4, the size of the general buffer considered in the experiments is a  $20 \times 50$  grid, where the total parking slots is 1,000, and the number of loading areas is 20. Each simulation run is continued until 20,000 vehicles complete their deployment, and a total of 20 simulation runs for each deployment model are conducted.



**Fig. 4.** Sizes of a general buffer and a loading buffer considered in simulations

Through the simulations, three deployment planning models are compared with respect to the selected performance measures, such as consolidated operational labor cost and shipment yard utilization. Three deployment models are: (i)  $DM^{\text{Current}}$ : current deployment model presented in Section 2.4, (ii)  $MA-DM^{\text{w/ODP}}$ : multiagent-based deployment model with the utility function  $u_i^{w/oDP}(b)$ , and (iii)  $MA-DM^{\text{wDP}}$ : multiagent-based deployment model with the utility function  $u_i^{wDP}(b)$ .

#### 4.1 Consolidated Operational Labor Cost

First, the average of consolidated operational labor cost per a vehicle is used to measure the performance of three deployment planning models. As explained in Section 2.2, consolidated a operational labor cost is computed based on the four elementary distances. It is obvious that the consolidated operational labor cost for a vehicle depends on which parking slot in a general buffer is allocated to the vehicle. For the simulation experiments, the delivery priority for each vehicle is randomly selected from a uniform random distribution of  $[0, 1]$ , the unit time labor costs for yard worker group A and B are given as 10 and 15 unit costs, and it is assumed that unit speeds of driving and riding are 10 and 8 times higher than that of walking.

**Table 1.** The average consolidated operational labor cost (COLC: unit cost) and the average walking distance (WD: unit distance) for a vehicle in different deployment planning models

	$DM^{Current}$		$MA-DM^{W/ODP}$		$MA-DM^{WDP}$	
	COLC	WD	COLC	WD	COLC	WD
Ave.	768.3	231.6	632.1	149.8	590.1	129.6

Table 1 shows the experimental results for the average consolidated operational labor cost and the average walking distance for a vehicle in three different models. The average consolidated operational labor costs in  $MA-DM^{W/ODP}$  and  $MA-DM^{WDP}$  are decreased as 17.7% and 23.2%, respectively, compared to the current deployment model. The average walking distances are reduced as 35.3% in  $MA-DM^{W/ODP}$  and 44.1% in  $MA-DM^{WDP}$  compared to the current model. From the results, it is indicated that the walking distance  $d_w(L, b)$  is a dominant factor in calculating the consolidated operational labor cost for a vehicle deployment. This lesson is quite reasonable in that walking from determined loading area  $L$  to slot  $b$  is the most time-taken operation in a vehicle deployment, because walking speed is much slower than driving and riding speeds.

#### 4.2 Shipment Yard Utilization

The consolidated operational labor cost can be reduced mainly by reducing the walking distance of a yard worker. This fact implies that increasing a utilization rate of the parking slot near loading areas results in the reduction of the walking distance. Through the analysis of experimental results, it is found that the utilization rate of the parking slot near loading areas is considerably increased in the proposed multiagent-based deployment model. Figure 5(a) and 5(b) show the utilization rates of parking slots in deployment planning with  $DM^{Current}$  and  $MA-DM^{WDP}$ , respectively. Deployment planning with  $MA-DM^{WDP}$  in the RTLS-enabled shipment yard significantly increases the utilization rate of the parking slots near loading areas compared to deployment planning with  $DM^{Current}$ . In this figure, parking slots with a lower column number represent they are located more close to loading areas than those with a higher column number. This fact provides an opportunity to reduce the size of a general buffer.

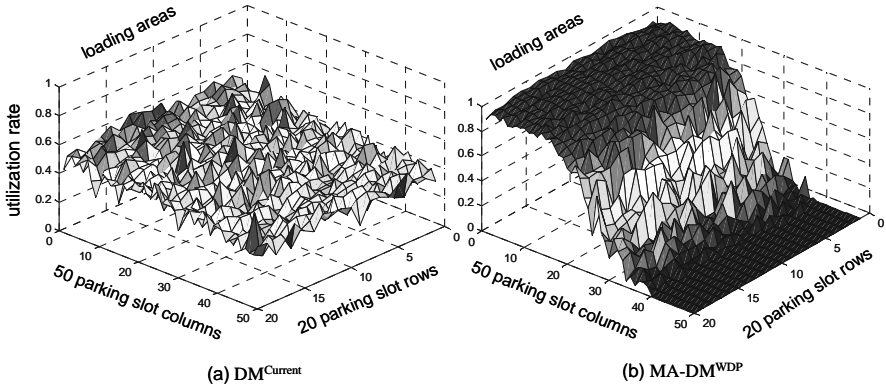


Fig. 5. The utilization rates of parking slots in (a)  $DM^{Current}$  and (b)  $MA-DM^{WDP}$

### 4.3 The Number (Frequency) of Vehicles Deployed into Parking Slots

As described in Section 3.4, the amount of time a vehicle with higher delivery priority stays in the general buffer is supposed to be shorter than that of a lower priority vehicle. To increase the number (frequency) of vehicles deployed to the parking slots near loading areas, the utility function with consideration of delivery priority is proposed in Section 3.4. As shown in Figure 6, the number of vehicles deployed into the parking slots near loading areas during the planning period are increased in  $MA-DM^{WDP}$  compared to  $MA-DM^{W/ODP}$ , which resulted in the reduction of the overall operational labor cost caused during the planning period. It is also obviously shown that the proposed multiagent-based deployment models considerably increase the number of vehicles deployed into the parking slots near loading areas.

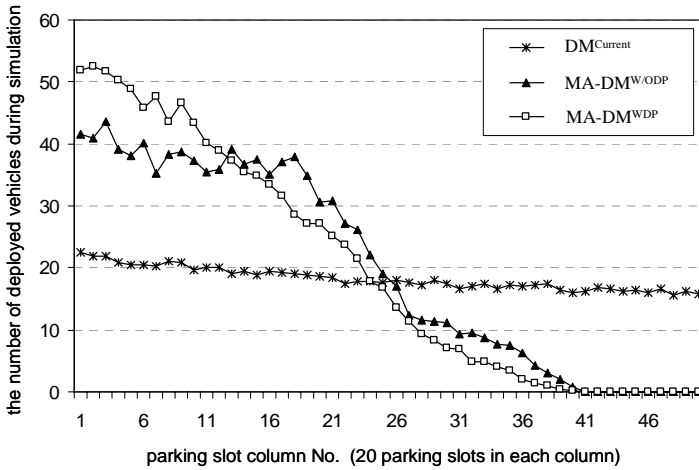


Fig. 6. The number of vehicles deployed into parking slots near loading areas

## 5 Concluding Remarks

This paper has presented a new approach to resolve the dynamic vehicle deployment planning problem in the RTLS-enabled shipment yard. The proposed approach uses a multiagent-based decentralized decision-making framework in which a market-based mechanism is facilitated to dynamically update initial vehicle deployment decisions. In this mechanism, a market is configured for determining a parking slot for each individual vehicle agent, and a utility function of a vehicle agent is designed in consideration of delivery priority.

The experimental results demonstrated that multiagent-based deployment planning models, MA-DM<sup>W/ODP</sup> and MA-DM<sup>WDP</sup>, outperformed DM<sup>Current</sup> with respect to the selected measures of deployment performance. The results also indicated that MA-DM<sup>WDP</sup> improved the utilization of the parking slots near loading areas, which leads to the reduction of overall operational labor cost. In addition, it was shown that the decision algorithms based on multiagent-based architecture and real-time vehicle information from RTLS can improve the performance further because these algorithms can capture the dynamics of the shipment yard environment in real-time.

## Acknowledgements

This study has been done with support from General Motors Research and Development Center through the research contract No. TSC82111.

## References

1. Angeles, R.: RFID Technologies: Supply-Chain Applications and Implementation Issues. *Information System Management*, 22 (2005) 51-65.
2. Bertsekas, D. P.: The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial. *Interfaces*, 20 (1990) 133-149.
3. Clearwater, S.: *Market-based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, Singapore (1996).
4. Datta, A., Viguiet, I. R.: Handling Sensor Data in Rapidly Changing Environments to Support Soft Real-Time Requirements. *Inform Journal on Computing*, 12 (2000) 84-103.
5. Kim, J., Kumara, S. R. T., Yee, S. -T., Tew, J.: Dynamic Shipment Planning in an Automobile Shipment Yard using Real-Time Radio Frequency Identification (RFID) Information. *Proceeding of the 2005 IEEE International Conference on Automation Science and Engineering*, Edmonton, Canada, August 1-2, (2005) 148-153.
6. Krothapalli, N. K. C., Deshmukh, A. V.: Design of Negotiation Protocols for Multi-Agent Manufacturing Systems. *International Journal of Production Research*, 37 (1999) 1601-1624.
7. Paqani, M.: *Mobile and Wireless Systems Beyond 3G: Managing New Business Opportunities*. IBM Press (2005).
8. Walker, W. T.: Emerging Trends in Supply Chain Architecture. *International Journal of Production Research*, 43 (2005) 3517-3528.
9. Wilding, R., Delgado, T.: RFID-Application within the Supply Chain. *Supply Chain Practice*, 6 (2004) 30-43.
10. Wu, T., Ye, N., Zhang, D.: Comparison of Distributed Methods for Resource Allocation. *International Journal of Production Research*, 43 (2005) 515-536.

# R-FRTDP: A Real-Time DP Algorithm with Tight Bounds for a Stochastic Resource Allocation Problem

Camille Besse, Pierrick Plamondon, and Brahim Chaib-draa

Computer Science & Software Engineering Dept., Laval University,  
Québec (Qc), Canada

{besse,plamon,chaib}@damas.ift.ulaval.ca  
<http://www.damas.ift.ulaval.ca/>

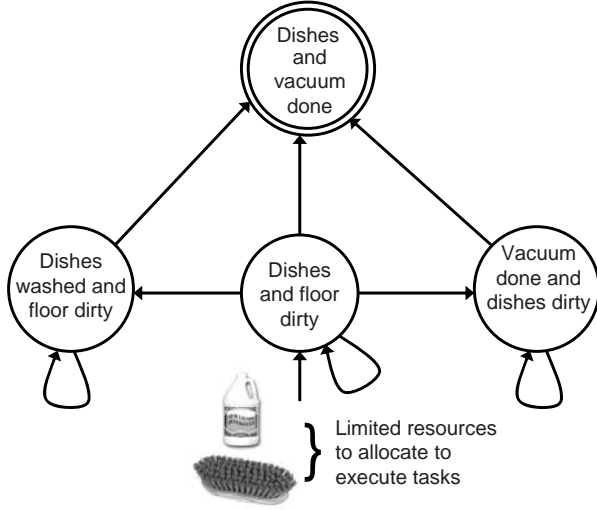
**Abstract.** Resource allocation is a widely studied class of problems in Operation Research and Artificial Intelligence. Specially, constrained stochastic resource allocation problems, where the assignment of a constrained resource do not automatically imply the realization of the task. This kind of problems are generally addressed with Markov Decision Processes (MDPs). In this paper, we present efficient lower and upper bounds in the context of a constrained stochastic resource allocation problem for a heuristic search algorithm called Focused Real Time Dynamic Programming (FRTDP). Experiments show that this algorithm is relevant for this kind of problems and that the proposed tight bounds reduce the number of backups to perform comparatively to previous existing bounds.

## 1 Introduction

Resource Allocation problem is a widely studied class of problem in Operation Research and Artificial Intelligence. This class is known to be NP-complete [10]. Since resources are usually constrained, the allocation of resources to one task restricts the options available for other tasks. As the action space is exponential according to the number of resources, and as the state space is exponential according to number of resources and tasks, this type of problem with time constraints is very complex.

A common way of addressing this large stochastic problem is by using Markov Decision Processes (MDPs), and in particular real-time search where many algorithms have been developed recently. For instance Real-Time Dynamic Programming (RTDP) [1], LRTDP [4], FRTDP [9], HDP [3], LAO\* [5] are all state-of-the-art heuristic search approaches in a stochastic environment. Because of its anytime quality, RTDP, introduced by Barto *et al.* [1] is an interesting approach since it updates states in trajectories from an initial state  $s_0$  to a goal state  $s_g$  in a very efficient way.

Actually, RTDP is much more effective if the action space can be pruned of sub-optimal actions. To do this, McMahan *et al.* [6], Smith and Simmons [9],



**Fig. 1.** Task transition graph

and Singh and Cohn [8] proposed solving a stochastic problem using a RTDP type heuristic search with upper and lower bounds on the value of states.

The bounds proposed by McMahan *et al.* [6] are related to a stochastic shortest path problem. Their approach is well suited for problems where the rewards (or costs) are obtained whenever an action is executed in a state. In the case of our resource allocation problem where rewards are only obtained when tasks are achieved, this approach is not applicable.

The FRTDP approach by Smith and Simmons [9] proposes an efficient trajectory of state updates to further speed up the convergence, given upper and lower bounds. This efficient trajectory of state updates are combined to the approach proposed in this paper as we focus on the definition of tight bounds for a constrained resource allocation problem.

On the other hand, the bounds proposed by Singh & Cohn [8] are suitable to our case, and extended in this paper using, in particular, the concepts of task criticality and feasibility to elaborate tight bounds. These bounds are implemented in the context of a FRTDP heuristic search approach.

Our bounds are compared theoretically and empirically to the bounds proposed by Singh & Cohn using the FRTDP approach. Indeed, when implementing FRTDP with our proposed upper and lower bounds, its convergence to the optimal policy is faster compared to the Singh & Cohn bounds using the same algorithm. Also, even if the algorithm used to obtain the optimal policy is RTDP, our bounds can be used with any other algorithm to solve an MDP. The only condition on the use of our bounds is to be in the context of stochastic constrained resource allocation.

Figure 1 gives an example of a stochastic resource allocation problem to execute tasks. In this problem, there are two tasks to realize:  $ta_1 = \{\text{wash the dishes}\}$ ,

and  $ta_2 = \{\text{clean the floor}\}$ . These two tasks are either in the realized state, or not realized state. Thus, the combination of the specific states of the individual tasks determines the four global states in Figure 1. To realize the tasks, two type of resources are assumed:  $res_1 = \{\text{brush}\}$ , and  $res_2 = \{\text{detergent}\}$ . A computer has to compute the optimal allocation of these resources to the cleaner robots to realize their tasks. In this problem, a state represents a conjunction of the particular state of each task and a time interval for which the task are going to be in this state. The resources may be constrained by the amount that may be used simultaneously. Furthermore, the higher is the number of resources allocated to realize a task, the higher is the expectation of realizing the task. A possible action in this state may be to allocate one unit of detergent to task  $ta_1$ , and one brush to task  $ta_2$ . The state of the system changes stochastically, as each task's state does. For example, the floor may be clean or not with a certain probability, after having allocated the brush to clean it.

## 2 Markov Decision Processes (MDPs) in the Context of Resource Allocation

A Markov Decision Process (MDP) framework is used to model our stochastic resource allocation problem. MDPs have been widely adopted by researchers today to model a stochastic process. This is due to the fact that MDPs provide a well-studied and simple, yet very expressive model of the world.

An MDP in the context of a resource allocation problem with limited resources is defined as a tuple  $\langle Res, Ta, S, A, P, W, R, \rangle$ , where:

- $Res = \langle res_1, ..., res_{|Res|} \rangle$  is a finite set of resource types available for a planning process.
- $Ta$  is a finite set of tasks with  $ta \in Ta$  to be executed.
- $S$  is a finite set of states with  $s \in S$ . A state  $s$  is a tuple  $\langle t_{start}, t_{end}, Ta, alloc \rangle$ . In particular,  $t_{start}$  is the start time of the state,  $t_{end}$  is the end time of the state.  $alloc$  is a set of allocations which are already in execution at time  $t_{start}$ . Also,  $S$  contains a non empty set  $s_g \subseteq S$  of goal states. A goal state is a sink state where an agent stays forever.
- $A$  is a finite set of actions (or assignments). The allocations  $a \in A(s)$  applicable in a state are the combination of all resource assignments that may be executed, according to the state  $s$ . In particular,  $a$  is simply an allocation of resources to the current tasks, and  $a_{ta}$  is the resource allocation to task  $ta$ . Each action has a start time  $t_{start}$  and an end time  $t_{end}$ . The possible actions are limited by the amount that may be used on a task at a particular time.
- Transition probabilities  $P_a(s'|s)$  for  $s \in S$  and  $a \in A(s)$ .
- $W = [w_{ta}]$  is the relative weight (criticality) of each task.
- State rewards  $R = [r_s] : \sum_{ta \in Ta} r_{s_{ta}} \leftarrow \Re_{s_{ta}} \times w_{ta}$ . The relative reward of the

state of a task  $r_{s_{ta}}$  is the product of a real number  $\Re_{s_{ta}}$  by the weight factor  $w_{ta}$ . For our problem, a reward of  $1 \times w_{ta}$  is given when the state of a task ( $s_{ta}$ ) is in an achieved state, and 0 in all other cases.

- A discount factor  $\gamma$ , which is a real number between 0 and 1. The discount factor describes the preference of an agent for current rewards over future rewards.

A solution of an MDP is a policy  $\pi$  mapping states  $s$  into actions  $a \in A(s)$ . In particular,  $\pi_{ta}(s)$  is the action (i.e. resources to allocate) that should be executed on task  $ta$ , considering the global state  $s$ . In this case, an optimal policy is one that maximizes the expected total reward for accomplishing all tasks. The optimal value of a state,  $V(s)$ , is given by:

$$V^*(s) = R(s) + \max_{a \in A(s)} \gamma \sum_{s' \in S} P_a(s'|s) V(s') \quad (1)$$

The end time  $t_{end}$  of  $s$  is set to the earliest ending time of an action in allocation ( $alloc$ ) or to execute ( $a$ ) in state  $s$ . The start time  $t_{start}$  of state  $s'$  is equal to time  $t_{end}$  of state  $s$ . Furthermore, one may compute the Q-Values  $Q(a, s)$  of each state action pair using the following equation:

$$Q(a, s) = R(s) + \gamma \sum_{s' \in S} P_a(s'|s) \max_{a' \in A(s')} Q(a', s') \quad (2)$$

where the optimal value of a state is  $V^*(s) = \max_{a \in A(s)} Q(a, s)$ . The policy is subjected to the resource constraint  $res(\pi(s)) \leq C_{s_{Ta}} \forall s \in S$ , and  $\forall res \in Res$ , where  $C_{s_{Ta}}$  is the resource constraint on tasks  $Ta$ . Heuristic search may reduce the complexity of a stochastic resource allocation problem by focussing on relevant states. To this end, Real-Time Dynamic Programming (RTDP) heuristic search is now introduced.

### 3 Real Time Dynamic Programming

Barto *et al.* [1] proposed Real Time Dynamic Programming (RTDP) (Algorithm 1) as an effective real-time heuristic search approach. RTDP is a simple dynamic programming algorithm that involves a sequence of trial runs, each starting in the initial state  $s_0$  and ending in a goal or a *solved* state. Each RTDP trial (TRIALRECURSE function) is the result of simulating the policy  $\pi$ , through the PICKNEXTSTATE function, while updating the upper bound values  $s.U$  using a Bellman backup (Equation 1) over the states  $s$  that are visited.  $h(s')$  is a heuristic which defines an initial value for state  $s'$ . This heuristic has to be admissible — The value given by the heuristic has to overestimate (or underestimate) the optimal value when the objective function is maximized (or minimized). For example, an admissible heuristic for a stochastic shortest path problem is the solution of a deterministic shortest path problem. Indeed, since the problem is stochastic, the optimal value is lower than for the deterministic version. The new set of tasks to accomplish is produced by the PICKNEXTSTATE function which randomly picks a none-*solved* state, containing a new set of tasks to realize, by executing the current policy.

It has been proven that RTDP, given an admissible initial heuristic on the value of states cannot be trapped in loops, and eventually yields optimal values [4].



**Algorithm 1.** RTDP

---

<b>Function</b> $\text{initNode}(s)$ : {implicitly called the first time each state $s$ is touched} 1: $s.U \leftarrow h_U(s)$ <b>Function</b> $\text{RTDP}(s_0, h_U)$ : 2: <b>loop</b> $\text{TRIALRECURSE}(s_0)$ <b>Function</b> $\text{BACKUP}(s)$ : 3: $s.U \leftarrow \max_a \text{QU}(s, a)$	<b>Function</b> $\text{TRIALRECURSE}(s)$ : 4: <b>if</b> $s \in \mathfrak{G}$ <b>then return</b> 5: $\pi(s) \leftarrow \arg \max_a \text{QU}(s, a)$ 6: $s' \leftarrow \text{s.PICKNEXTSTATE}()$ 7: $\text{TRIALRECURSE}(s')$ 8: $\text{BACKUP}(s)$ <b>Function</b> $\text{QU}(s, a)$ : 9: <b>return</b> $R(s, a) + \gamma \sum_{s' \in \mathcal{S}} \gamma T_{s,s'}^a s'.U$
--	---

---

**3.1 Focused RTDP**

Focused RTDP (alg. 2) is an RTDP based algorithm proposed by Smith & Simmons [9]. As in RTDP, FRTDP's execution consists in trials that begin in a given initial state  $s_0$  and then explore reachable states of the state space, selecting actions according to an upper bound. Once a final state is reached, it performs Bellman updates on the way back to  $s_0$ .

Unlike RTDP, FRTDP maintains also a lower bound and uses others criteria to select actions outcomes and to detect trial termination. The lower bound is used to establish the policy and it also contributes in the priority calculation of states to expand on the fringe of the search tree. Trial termination detection has been modified and improved by adding an adaptive maximum depth  $D$  in the search tree in order to avoid over-committing to long trials early on. In fact, the maximum depth  $D$  is updated by  $k_D D$  each time the trial is not useful enough. This usefulness is represented by  $\delta W$  where  $\delta$  measures how much the update changed the upper bound value of  $s$  and  $W$  the expected amount of time the current policy spends in  $s$ , adding up all possible paths from  $s_0$  to  $s$ . Refer to the pseudo-code of Algorithm 2 and to Smith & Simmons' article [9] for details.

**3.2 Singh & Cohn's Lower and Upper Bounds**

Singh & Cohn [8] defined lower and upper bounds for a stochastic problem. Their approach is pretty straightforward. First of all, a value function is computed for all tasks to realize, using a standard RTDP approach. Then, using these *task*-value functions, a lower bound  $h_L$ , and upper bound  $h_U$  can be defined. In particular,  $h_L(s) = \max_{ta \in T_a} V_{ta}(s_{ta})$ , and  $h_U(s) = \sum_{ta \in T_a} V_{ta}(s_{ta})$ . The admissibility of these bounds has been proven by Singh & Cohn, such that, the upper bound always overestimates the optimal value of each state, while the lower bound always underestimates the optimal value of each state. In this paper, the bounds defined by Singh & Cohn and implemented using FRTDP define the SINGH-FRTDP approach.

The next sections propose to tighten the bounds of SINGH-FRTDP to permit a more effective pruning of the action space.

**Algorithm 2.** Focused RTDP

---

**Function** INITNODE( $s$ ): {implicitly called the first time each state  $s$  is touched}

1:  $(s.L, s.U) \leftarrow (h_L, h_U)$ ;  $s.prio \leftarrow \Delta(s)$

**Function** FRTDP( $s_0, \varepsilon, h_L, h_U, D_0, k_D$ ):

2:  $D \rightarrow D_0$

3: **while**  $s_0.U - s_0.L > \varepsilon$  **do**

4:  $(q_p, n_p, q_c, n_c) \leftarrow (0, 0, 0, 0)$

5: TRIALRECURSE( $s_0, W = 1, d = 0$ )

6: **if**  $(q_c/n_c) \geq (q_p/n_p)$  **then**

$D \leftarrow k_D D$

7: **end while**

**Function** TRIALRECURSE( $s, W, d$ ):

8:  $(a^*, s^*, \delta) \rightarrow \text{BACKUP}(s)$

9: TRACKUPDATEQUALITY( $\delta W, d$ )

10: **if**  $\Delta(s) \leq 0$  **or**  $d \geq D$  **then return**

11: TRIALRECURSE( $s^*, \gamma T_{s,s^*}^{a^*} W, d + 1$ )

12: BACKUP( $s$ )

**Function** TRIALUPDATEQUALITY( $q, d$ ):

13: **if**  $d > D/k_D$  **then**

$(q_c, n_c) \leftarrow (q_c + q, n_c + 1)$

14: **else**  $(q_p, n_p) \leftarrow (q_p + q, n_p + 1)$

---

**Function** BACKUP( $s$ ):

15:  $s.L \leftarrow \max_a \text{QL}(s, a)$

16:  $u \leftarrow \max_a \text{QU}(s, a)$

17:  $a^* \leftarrow \arg \max_a \text{QU}(s, a)$

18:  $\delta \leftarrow |s.U - u|$

19:  $s.U \leftarrow u$

20:  $p \leftarrow \max_{s' \in \mathcal{S}} \gamma T_{s,s'}^{a^*} s'.prio$

21:  $s^* \leftarrow \arg \max_{s' \in \mathcal{S}} \gamma T_{s,s'}^{a^*} s'.prio$

22:  $s.prio \leftarrow \min(\Delta(s), p)$

23: **return**  $(a^*, s^*, \delta)$

**Function**  $\Delta(s)$ :

24: **return**  $|s.U - s.L| - \varepsilon/2$

**Function** QL( $s, a$ ):

25: **return**  $R(s, a) + \gamma \sum_{s' \in \mathcal{S}} \gamma T_{s,s'}^a s'.L$

**Function** QU( $s, a$ ):

26: **return**  $R(s, a) + \gamma \sum_{s' \in \mathcal{S}} \gamma T_{s,s'}^a s'.U$

---

**3.3 Reducing the Upper Bound**

In the next two sections, tight bounds are proposed for a stochastic resource allocation problem. The FRTDP approach initiated with these bounds defines the Resource FRTDP (R-FRTDP) approach.

The upper bound of SINGH-FRTDP includes actions which may not be possible to execute because of resource constraints. To consider only possible actions, the upper bound is now:

$$h_U(s) = \max_{a \in A(s)} \sum_{ta \in T_a} Q_{ta}(a_{ta}, s_{ta}) \quad (3)$$

where  $Q_{ta}(a_{ta}, s_{ta})$  is the Q-value of task  $ta$  for state  $s_{ta}$ , and action  $a_{ta}$  computed using a FRTDP approach.

When the time is introduced into the problem, matching task states to the global state is not obvious. Indeed, the start time and end time of a global state are generally different of the start time and end time of the specific state of each task. This is caused by the fact that the end time of a state  $s$  is obtained according to the time of the first action to end when considering all tasks. On the other hand, the end time of a task state  $s_{ta}$  is obtained with the first action to end, when considering the current task  $ta$  only.

To match correctly a task state  $s_{ta}$  within a global state  $s$ , we find a task state for which:

$$t_{start_{ta}} \leq t_{start} < t_{end_{ta}} \quad (4)$$

where  $t_{start_{ta}}$  and  $t_{end_{ta}}$  are, respectively the starting and ending time of  $s_{ta}$ . Also,  $s_{ta}$  has to match the other characteristics of the task  $ta$  in the global state  $s$ .

Here, the best matching state for a task is found. It is a matching state since the start time of  $s_{ta}$  is less or equal than the start time of the global state  $s$ . Indeed, the possible actions in state  $s_{ta}$  are equal or greater than the possible actions in state  $s$  for task  $ta$ . Also, it is the *best* matching state because one and only one state for a task can satisfy Equation 4 and it is the state which has the start time the nearest possible of  $t_{start}$ , but for which  $t_{start_{ta}}$  is not greater than  $t_{start}$ .

**Theorem 1.** *The upper bound of Equation 3 is admissible.*

**Proof:** The resource constraints are satisfied because the upper bound is computed using all global possible actions  $a$ . However,  $h_U(s)$  still overestimates  $V^*(s)$  because the future states of the tasks violates the resource constraint. Indeed, each task may use all consumable resources for its own purpose. Doing this produces a higher value for each task, than the one obtained when planning for all tasks globally with the shared limited resources. ■

**Complexity of Computing the Upper Bound.** For the upper bound, SINGH-FRTDP and R-FRTDP compute a value function for each task. We consider  $|S_{ta}|$  as the number of possible states for task  $ta$ . Also,  $|S_{Ta}|$  is the number of possible joint states for all tasks  $ta \in Ta$ . Since  $|S_{Ta}|$  is combinatorial with the number of tasks, thus  $|S_{ta}| \ll |S_{Ta}|$ . Indeed,

$$|S_{Ta}| = \mathcal{O}(|S_{ta}|^{|Ta|}) \quad (5)$$

When the number of tasks is high, the complexity of computing a value function for each task is negligible compared to computing a global value function for all tasks. The main difference in complexity between the SINGH-FRTDP approach, and R-FRTDP is how the value function is used. The SINGH-FRTDP approach simply sums the value function  $V_{ta}(s_{ta})$  of each task  $ta$  to determine the upper bound of a state  $s$ . As for R-FRTDP, all global actions  $a \in A(s)$  are computed to determine the maximal possible upper bound, considering the resource constraints of a state  $s$ . Thus, the complexity to determine the upper bound of a state is  $\mathcal{O}(|A| \times |Ta|)$ . The computation of all global actions is much more complex than simply summing the value functions, as in SINGH-FRTDP. A standard Bellman backup, when computing the global solution sums  $|S|$  for each  $a \in A(s)$ , thus has complexity  $\mathcal{O}(|A| \times |S|)$ . Since  $|A| \times |Ta| \ll |A| \times |S|$ , the computation time to determine the upper bound of a state, which is done one time for each visited state, is much less than the computation time required to compute a standard Bellman backup for a state, which is usually done many times for each state. Thus, the computation time of the upper bound is negligible.

**Increasing the Lower Bound.** As time is integrated in this problem a first heuristic is to use a rule based reactive planner to assign resources as tasks income. Thus, the lower bound we develop assign resources to tasks according to their priority and remaining time while respecting constraints as described by algorithm 3. Practically, trials are made with action chosen by the reactive planner in order to evaluate its policy. Choice of next states of actions are made depending on the same criteria as FRTDP (see section 3.1 or [9]). An adaptive number of trial is also chosen, until the value gained between each trial is no more a fixed threshold. As a result, an approximation of a sub-optimal policy is calculated depending on time wanted to be spend on lower bound calculation.

**Theorem 2.** *The lower bound proposed in this section is admissible.*

**Proof:** This is a lower bound since the reactive policy is sub-optimal and resource constraints are checked by the algorithm 3. ■

---

**Algorithm 3.**  $h_L$ : Reactive algorithm [2]

---

```

1: Inputs: Tasks: Tasks list;
2: Resources: Resources list;
3: {Tasks pre-treatment:}
4: Tasks  $\leftarrow$  PRIORITIZE(Tasks)
5: Resources  $\leftarrow$  ORDERBYEFFICIENCY(Resources)
6:  $T \leftarrow$  FIRST(Tasks)
7:  $R \leftarrow$  FIRST(Resources)
8: while Resources  $\neq \emptyset$  do
9:   if AVAILABLE( $R$ ) and ASSIGNABLE( $R, T$ ) then
10:    ASSIGN( $R, T$ )
11:    Tasks  $\leftarrow$  Tasks  $\setminus \{T\}$ 
12:    Resources  $\leftarrow$  Resources  $\setminus \{R\}$ 
13:     $T \leftarrow$  FIRST(Tasks)
14:   else
15:     $T \leftarrow$  NEXT(Tasks)
16:   end if
17:    $R \leftarrow$  FIRST(Resources)
18: end while
19: return An allocation of all available resources

```

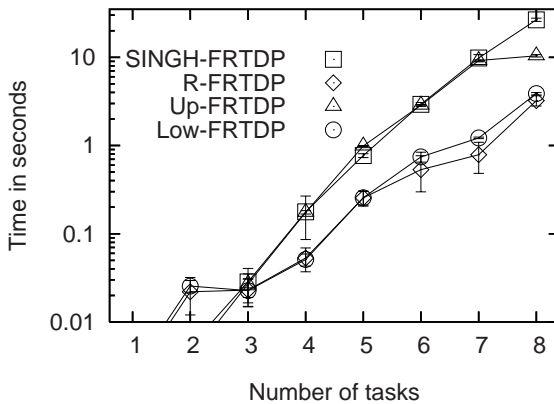
---

## 4 Experimental Results

The domain of the experiments is a naval platform which must counter incoming missiles (i.e. tasks) by using its resources (i.e. weapons, movements). For the experiments, 100 randomly resource allocation problems were generated for each approach, and possible number of tasks. In our problem,  $|S_{ta}|$  was generally 7, thus each task can be in 6 distinct states. In particular, there were generally

5 possible states for a missile where actions can be performed to counter it. In other words, there are 5 possible reallocation for a missile until it is too late to execute it. The number of reallocation depends greatly on the speed of the missile and its time of appearance. For each task, there are two goal states; a state where the missile is realized, and a state where the missile has hit the ship. The state transitions are all stochastic because when a missile is in a given state, it may always transit in many possible states. The number of resource type has been fixed to 3, where each type has constraints on the amount that may be used at a time. In particular, at most 1 resource of any type can be allocated on a task on a particular time. This constraint is also present on a real naval platform because of sensor and launchers constraints and engagement policies. Each resource type has its specific range of effectiveness. In particular, the first resource type generally has 3 possible reallocations before the threat is too near of the ship to be able to use this resource. The effectiveness of this resource varies between 90% and 95%. These variations in effectiveness depends greatly on the type of threat and its range from the ship. The second resource type has a part of its effectiveness range which overlaps with the first resource type and has 2 possible reallocations on a threat. The effectiveness of this resource is between 35% and 50%. The last resource type can be used when the threat is very near the ship and has no possible reallocation to it. The effectiveness of this resource varies between 65% and 85%. The bulk of the planning work is on made on the decisions of how to allocate the first resource type which has a big range and a high probability of effectiveness.

The optimal R-FRTDP, SINGH-FRTDP, UP-FRTDP and LOW-FRTDP approaches are compared in Figure 2. Two more versions have been added to the results. First of all, UP-FRTDP uses the lower bound of Singh & Cohn [8] and the upper bound of Section 3.3. Then, LOW-FRTDP uses the upper bound of Singh & Cohn [8] and the lower bound of Section 3.3.



**Fig. 2.** Computational efficiency of R-FRTDP, SINGH-FRTDP, UP-FRTDP (Lower bound of Singh & Cohn and upper bound of Section 3.3), LOW-FRTDP (upper bound of Singh & Cohn and lower bound of Section 3.3)

In terms of experiments, notice that the SINGH-FRTDP approach for resource allocation, which use loose bounds requires the most time for convergence. For instance, it takes an average of 26.6 seconds to plan for an SINGH-FRTDP approach with eight tasks (see Figure 2). The R-FRTDP approach solves optimally the same type of problem in an average of 3.23 seconds. This is a very significant improvement. The number of iterations required for convergence is significantly smaller for R-FRTDP SINGH-FRTDP. Indeed, the more tight the bounds are, the faster these bounds converge to the optimal value.

On the figure results, we may also observe that the reduction in planning time of R-FRTDP compared to SINGH-FRTDP is obtained mostly with the lower bound. Indeed, when the number of task to execute is high, the lower bounds by SINGH-FRTDP takes the values of a single task. On the other hand, the lower bound of R-FRTDP takes into account the value of all task by using a heuristic to distribute the resource. Indeed, an optimal allocation is one where the resources are distributed in the best way to all tasks, and our lower bound heuristically does that.

## 5 Conclusion

The experiments have shown that R-FRTDP provides a potential solution to solve efficiently stochastic resource allocation problems. Indeed, the planning time of R-FRTDP is significantly lower than for FRTDP with no initial heuristic or with the Singh & Cohn [8] heuristic. While the theoretical complexity of R-FRTDP is higher than for SINGH-FRTDP, its ability to produce a tight bound offsets this aspect, as shown in the experiments.

An interesting research avenue would be to experiment R-FRTDP with other heuristic search algorithms than FRTDP. HDP [3], and LAO\* [5] are both efficient heuristic search algorithms which could be implemented using our bounds. As a matter of fact, the bounds proposed in this paper can be used with any stochastic algorithm which solves a perfectly observable resource allocation problem.

Furthermore, our approach could be improved by considering the probability of new tasks coming in the environment and reserving resources for them as done by Mercier & Van Hentenryck [7]. This would permit to have a more effective model and thus a better allocation.

## References

1. A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.
2. D. Blodgett, P. Plamondon, B. Chaib-draa, P. Kropf, and E. Boss. A method to optimize ship maneuvers for the coordination of hardkill and softkill weapons within a frigate. In *7<sup>th</sup> International Command and Control Research and Technology Symposium (7<sup>th</sup> ICCRTS)*, Quebec, QC, September 2002.
3. B. Bonet and H. Geffner. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, August 2003.

4. B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proceeding of the 13Th International Conference on Automated Planning & Scheduling (ICAPS-03)*, pages 12–21, Trento, Italy, 2003.
5. E. A. Hansen and S. Zilberstein. LAO\* : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.
6. H. B. McMahan, M. L., and G. J. Gordon. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 569–576, New York, NY, USA, 2005. ACM Press.
7. L. Mercier and P. V. Hentenryck. Performance analysis of online anticipatory algorithms for large multistage stochastic integer programs. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.
8. S. Singh and D. Cohn. How to dynamically merge markov decision processes. In *Advances in neural information processing systems*, volume 10, pages 1057–1063, Cambridge, MA, USA, 1998. MIT Press.
9. T. Smith and R. Simmons. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston, USA, 2006.
10. W. Zhang. Modeling and solving a resource allocation problem with soft constraint techniques. Technical report: WUCS-2002-13, Washington University, Saint-Louis, Missouri, 2002.

# A Reorganization Strategy to Build Fault-Tolerant Multi-Agent Systems

Sehl Mellouli

Département of Organizational Information Systems,  
Laval University, G1K 7P4, Quebec, Canada  
`sehl.mellouli@sio.ulaval.ca`

**Abstract.** MAS failures are not only due to programming exceptions; they may originate from other sources such as the environment of the MAS which may influence the MAS' behavior. Furthermore, MAS fault-tolerant techniques based on agent replication cannot always be applied to a MAS. For example, it is not always possible to replicate a costly robot in a robotic MAS application. In this paper, we propose a reorganization strategy, based on *both* task and agent replication, to enable a MAS to detect and recover from its failures. Our strategy is different from those presented in the literature, which are based on agent replication, since it does not deal with programming faults but with failures originating from the MAS environment, and it is based on task and agent replication and not only on agent replication. Our strategy is scalable and is robust in detecting agents failure.

## 1 Introduction

Several fault-recovery techniques, based on agent replication, have been proposed in the literature to build fault-tolerant multi-agent systems [2] [3] [4] [5] [6] [8] [9] [11]. However, considering the case of a robotic MAS, it is not always possible to replicate costly robots. Consequently, these fault-tolerant techniques could not be applied to a MAS in this case. Furthermore, a MAS may have several sources of failures originating from programming faults as described in [6]. However, a MAS may operate under conditions which may prevent it from achieving its tasks. For example, if a robot that must move is blocked by an obstacle which it cannot encounter, then there should be another robot in the system to carry out the tasks of the failed one. If not, the MAS is in failure without having programming exceptions. In this paper, we will not address issues related to programming failures but to other sources of failures which may prevent the MAS from achieving its tasks, those induced by the environment.

In this paper, we present a reorganization strategy which includes:

- a fault-recovery technique that detects the existence of faults and eliminate them;
- a fault-tolerance technique that provides services complying with the specifications of the system in case of faults.



This strategy is evaluated with regard to its scalability, to the required time for the system to recover from its failures, to the required time to detect an agent failure, and to its robustness to detect agents failure. As presented in Section 4, the technique is scalable and robust.

The paper is organized as follows. Section 2 proposes a categorization of failures. Section 3 describes the reorganization strategy. Section 4 presents the experiment results. Section 5 compares our work with related work on fault-tolerance techniques. Section 6 concludes this paper.

## 2 Categorization of Agent Failures

In this paper, we do not address failures originating from programming exceptions. We address failures originating from other sources mainly the environment. We try to answer the question: what could be these other sources of failure?

To this end, we start from the assertion that a MAS is situated within an environment with which it interacts. The environment of a MAS is defined as what lies outside the MAS' boundary [12]. The MAS' boundary is the area within which the MAS can make things happen, or prevent them from happening. Consequently, if the MAS is in failure, then it will be able to prevent this failure from happening if the failure is originating from the MAS' boundary. However, the MAS cannot prevent failures originating from its environment; we address this kind of failures. When developing critical fault-tolerant systems such as security systems, the different possible system failures, either those originating from program failures or those originating from the environment must be taken into account in order to overcome them when they occur.

The sources of failures we address here can be thought of as preconditions to execution of agents' tasks [1]. These preconditions may influence the whole MAS, the agents, or the tasks. They can be grouped into three categories. The preconditions in the first category are those that, if not met, prevent the MAS from being operational. For example, a group of robots moving on a solar planet can see its activities shutdown if the conditions on this planet are unfavorable (like a solar storm). The preconditions of the second category are those that, if not met, prevent individual agents from being operational, as for example when one robot is out of electric power. The preconditions of the third category are those that, if not met, prevent agent tasks from being executable. For example, if a robot moves and encounters an obstacle which it is not able to avoid, then it will no longer move. In this case, one of the tasks of the robot cannot be performed. However, the robot may continue providing services to other robots which do not require that the robot moves. Table 1 summarizes these three categories of failures.

Furthermore, each precondition of either category 2 or category 3 may affect respectively only one agent (or task) or several agents (or tasks). This categorization will be used in Section 4 to determine when the MAS may be able to recover from its failures or not, and if so, how.

In this section, we provided a new categorization of MAS failures which should be taken into account when developing a MAS.

**Table 1.** Agents failure categories

Failure type	Description
Category 1: MAS failure	This type of failure prevents the MAS from functioning
Category 2: Agent failure	Any failure of this type prevents a set of agents from functioning
Category 3: Task Failure	Any failure of this type prevents individual task from functioning

### 3 The Reorganization Strategy

As presented in [13], at least the following techniques should be taken into account in order to build fault-tolerant systems :

1. *Fault-prevention* technique: to prevent fault introduction and occurrence;
2. *Fault-recovery* technique: to detect the existence of faults and eliminate them;
3. *Fault-tolerance* technique: to provide services complying with the system's objectives in case of faults.

The reorganization strategy we present in this paper includes a fault-recovery technique, and a fault-tolerance technique. The fault-prevention technique will be developed in future work. In what follows, we present these two techniques. Before introducing these technique, we present in the next section the different concepts required to define them.

#### 3.1 Basic Concepts

Let  $A$  be the set of agents operating in a MAS. If there are  $m$  agents,  $A = \{a_i, i \in [1..m]\}$ ;  $|A| = m$ . Each agent can perform several tasks. Tasks are the activities required, or believed to be necessary for an agent to achieve a goal in an interactive environment [1]. We define  $\tau$  the set of all tasks to be performed by agents. Let  $|\tau| = q$ , and  $\tau = \{t_i, i \in [1..q]\}$ . Each task has a set of preconditions which must be met so that the task is properly performed. These preconditions are in category 3. Also, each task has post-conditions which are conditions that must be met after the execution of the task.

For a MAS to operate correctly, its agents may require to use resources. If a resource, required by the system, is not available, then the system may be in failure. Hence, the set of resources required by the system must be identified. We define, in what follows, the set  $R$  of resources required by the system;  $R = \{r_1, \dots, r_l\}$ .  $|R| = l$ .

In this section, we presented the basic concepts on which our reorganization is defined. In what follows, we present the reorganization technique.

### 3.2 The Fault-Tolerance Technique

Our *fault-tolerance* technique is based on task and agent redundancy. The idea is to first replicate tasks whenever possible, and if not, then to replicate agents. Hence, the fault-tolerance technique should reduce the overall MAS complexity by minimizing the number of agents to be replicated.

In a MAS, there are agents determined at system design which are called *original agents*, and there are agents added to the system to replicate original agents in order to acquire the system with fault-tolerance capabilities which are called *replica agents*. Consequently, a fault-tolerant multi-agent system will be composed of the original agents and of the replica of some agents. Let  $O_a$  be the set of the original agents of a MAS. Let  $R_a$  be the set of the replica agents. The set of agents  $A$  is defined as the union of  $O_a$  and  $R_a$ .

As stated before, our approach is based on first replicating tasks if possible, otherwise replicating original agents. To determine which tasks and original agents to replicate, we introduce the notions of *critical* and *non-critical agents*. A *critical agent* is an original agent which performs at least one task that cannot be replicated in any other original agent in the system. An original agent task could be replicated in any other original agent if the latter has access to the required resources of that task. A *non-critical agent* is an original agent for which each task can be replicated in at least another original agent. We propose to replicate all the *critical agent*. All the *non-critical agents* will see their tasks replicated in other *original agents*. Each task is replicated in only one agent. If there are two possible agents in which a given task can be replicated, then the MAS' designer may use several criteria in order to choose the agent in which to replicate the tasks. These criteria can depend, for example, on the cost to replicate a task in another agent or to replicate a task into the agent which has a minimal number of tasks to perform. This problematic is out of the scope of this paper.

Now, when the system will operate, agents must identify the agents in which their tasks are replicated in order to coordinate their activities. To this end, we couple the agents on a task basis so that we avoid the overhead associated with task reallocation after a task failure. The coupling function is implemented in the *Task* class and returns the list of agents which handle this task. Each task maintains a list of the two agents which can perform it (the original one and the replica). When an agent is created, it is assigned a set of tasks, and each assigned task will see its list of agents handling it updated.

Now that our fault-tolerance technique is defined, the MAS should recover from its failures. Before introducing our fault-recovery technique, we determine the situations from which the MAS may recover and those from which it cannot. If a precondition of category 1 is not met then the MAS cannot operate; it is in a fatal failure. In this case, it is not possible for the MAS to recover from its failure except if there is an external intervention to make this precondition met.

If at least one precondition of category 2 is not met then there is at least an agent  $a_1$  down. If all the not met  $a_1$ 's preconditions are not shared with  $a_1$ 's replica, then the replica may take over the failed agent and the MAS may recover

from its failure. Otherwise, the replica will be in failure too and the MAS would not be able to recover from its failure.

If at least one precondition of category 3 is not met, or one of the required resources is not available then there is at least a task  $t_1$  in failure. If all the not met preconditions or not available  $t_1$ 's resources are not shared with a  $t_1$ 's replicated task (which is implemented in another agent) then the MAS should be able to recover from its failure since  $t_1$ 's replicated task should be performed. Otherwise, it will not recover from its failures.

Now that we identified the cases in which the MAS may recover from its failures, we present in the next section the fault-recovery technique.

### 3.3 The Fault-Recovery Technique

The *fault-recovery* technique should allow the MAS to continue operating correctly despite failures whenever possible as stated earlier. It should allow the system to first detect failures and then recover from them. In this section, we provide techniques to detect and recover from failures. To illustrate our techniques, we consider an agent  $a_i$  in failure such that its tasks are replicated in more than one agent.

#### *Fault Detection Techniques*

Each agent should be able to detect other agents' failure and help to recover from it. Let  $a_j$  be an agent such that  $a_i$  has some of its tasks replicated in  $a_j$ , i.e., each agent maintains a list of agents with which it communicates.<sup>1</sup>  $a_i$  can be down or some (and not all) of its tasks could be in failure. In order for the agents in  $a_i.duplicate()$  to take over the failed tasks, they have to know whether  $a_i$  is down or in failure. If  $a_i$  is in failure and cannot perform a task  $t$ , then it should notify a counterpart, an other agent  $a_j$  acting as a backup agent for task  $t$ , i.e.,  $a_j$  is coupled with  $a_i$  for task  $t$ . Otherwise, it wouldn't be able to inform the other agents. To that effect we propose to use a handshake protocol in order to ensure that coupled agents know whether their counterparts are down or not. In Section 4, we evaluate the impact of the number of agents operating in the system on the number of handshake messages exchanged between agents.

If  $a_i$  is down, then each of the agents in  $a_i.duplicate()$  will not receive a handshake from  $a_i$ , and will assume that they have to take over. Nevertheless, all these agents must agree that  $a_i$  is down in order to take over. We propose to use a counter on the number of agents agreeing that  $a_i$  is down. All the agents which are in  $duplicate(a_i)$  can modify the value of this counter. Each agent which detects that  $a_i$  is down increments the value of this counter. If the value of the counter is equal to  $|a_i.duplicate()|$ , then the agents agree that  $a_i$  is down. Otherwise, the agents that have incremented the value of the counter deduce that  $a_i$  is not able to communicate with them, but  $a_i$  is not down. Furthermore, there could be a loss of handshake messages in the MAS which may lead to false failure detection; that is agents detect that an agent is down but it is not

---

<sup>1</sup> A duplicate function is implemented in the agent class which returns, for each agent instance, the list of agents with which it communicates.

the case. In Section 4, we test the robustness of our technique with regard to handshake messages loss.

The algorithm defining the fault-detection is presented in what follows.

Let  $L$  be the set of agents with which an agent  $a_i$  has to communicate;

Let *ldownAgent* be a list of agents which may be down;

```

for  $\forall a_j \in L$  do
  if  $a_i.\text{handShake}(a_j) = \text{false}$  then
    Add  $a_j$  to ldownAgent;
    let lAgentCommunicateWith be the set of agents in which  $a_j$  tasks are
    replicated;
    for  $\forall a \in \text{lAgentCommunicateWith}$  do
      if  $a.\text{handShake}(a_j) = \text{false}$  then
        counter := counter + 1; – we increment the counter associated to  $a_j$ .
      end if
    end for
  end if
end for
if counter = lAgentCommunicateWith.getSize() then
  Agent  $a_j$  is down;
end if
end for

```

Now that a failure is detected, the system should recover from it, whenever possible.

#### *Fault Recovery Technique*

In what follows, we present how the system recovers from its failures. Let  $a_i$  be an agent in failure. Each task  $t$ , of  $a_i$ , in failure will be performed by agent  $a_j$  in  $a_i.\text{duplicate}()$  such that  $a_i$  and  $a_j$  are coupled for  $t$ . Nevertheless,  $a_j$  should have a copy of  $a_i$ 's knowledge so that it can continue operating from the last non-faulty point, if possible.  $a_j$  can have a copy of  $a_i$ 's knowledge by receiving  $a_i$ 's knowledge from it and in which  $a_i$ 's knowledge pertaining to task  $t$  is stored.

This knowledge is implemented as an abstract class (*Knowledge.java*) in which there is a reference to the agent owning this knowledge. In addition, each agent has an attribute of type *Knowledge* which represents the mental state of the agent. Each agent maintains a copy, as a list, of the mental states of the agents for which it replicates their tasks. Doing so, when an agent mental state of this list will be updated, we have only to update its corresponding element of the list.

When an agent is in failure, each of its tasks in failure will be performed by replica agents. Consequently, the communication links between agents will change since the agents that were communicating with the agent in failure will now communicate with some of its replica. The time needed for the agent to change their communication links is evaluated in Section 4.

The algorithm for changing agent communication links is described in what follow:

**if** an agent is down **then**

Let  $L$  be the list of all agents needing to communicate with the down agent;

**end if**

**for**  $\forall a_k \in L$  **do**

Let  $sbl$  be the set of tasks to be performed by the down agent;

**for**  $\forall t \in sbl$  **do**

Determine a back up agent  $a'$  to perform  $t$ ;

Add  $a'$  to the set of agents with which  $a_k$  communicate;

**end for**

**end for**

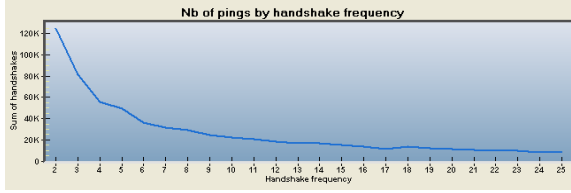
To summarize, the *fault-recovery* technique is based on:

- fault detection: faults are detected based on a presence notification mechanism that relies on:
  - a) a direct handshake protocol between coupled agents;
  - b) a request for the counterpart agent whenever a task failure is detected by an agent.
- fault recovery: the system can recover from its failures based on the shared memory areas between agents.

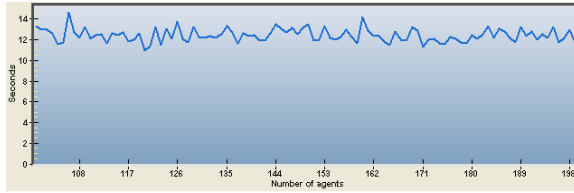
## 4 Experiments and Results

We presented a reorganization strategy which may allow the MAS to recover from its failures. Our strategy is based on a handshake protocol. However, there is a risk of losing handshake messages and consequently a false agent failure detection may occur. In addition, when an agent is in failure, new communication links must be established between agents. In brief, we have to evaluate our reorganization strategy by the following tests:

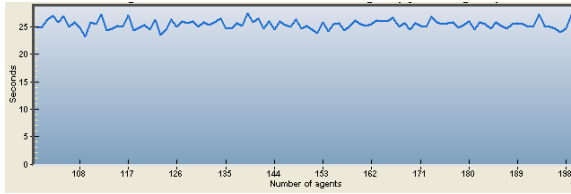
1. the first test evaluates the number of handshake messages exchanged between a 200 agents system (see Figure 1) while changing the time frequency of these messages from 2 seconds to 25 seconds;
2. the second test evaluates the time taken to change the communication links between agents in case of failure. The test is based on a number of agents varying from 100 to 200, while increasing the number of tasks per agent from two tasks per agent (see Figure 2) to 20 tasks per agent (see Figure 3);
3. the third test provides the rate of false failure detection when messages are lost. The test is based on a number of agents varying from 40 to 200 (see Figures 4 and 5) with a rate of lost messages varying from 20% to 100%. The rate of false failure detection is evaluated as the ratio of the number of false failure detection by the number of all failure detection. The number of all failure detection is the sum of the number of real failure detection and the number of false failure detection;
4. the fourth test is the mean time for detecting errors in the MAS by handshake frequency (see Figure 6).



**Fig. 1.** Number of handshake messages with 200 agents



**Fig. 2.** Required time for changing communication links with 2 tasks per agent

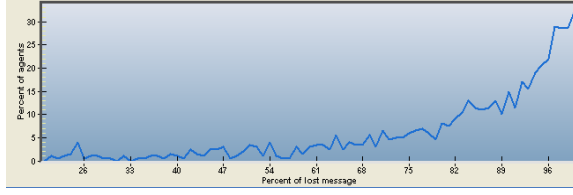


**Fig. 3.** Required time for changing communication links with 20 tasks per agent

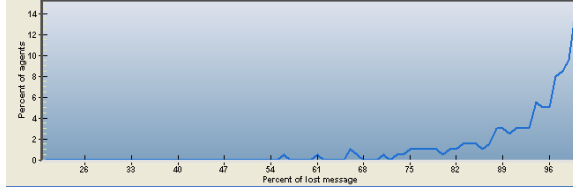
The different tests have been done on a 1.6GHz pentium processor with a 512MO RAM. Each test was run 50 times then we evaluated the mean value of all these tests as depicted in the different graphics. Furthermore, in order to make the tests more realistic, we changed the values resulting from the evaluation of the preconditions at run time in order to randomly bring agents up or down. Each agent performs 20 tasks while in the second test, we have experiments with agents only performing two tasks.

For the first test (Figure 1), we can see that as the time interval between handshake messages increases, the number of messages exchanged between agents considerably decreases, which is an expected result. Consequently, depending on the capacity of the network hosting the MAS, we determine the handshake messages frequency. If the network offers a high bandwidth capacity then we can use short handshake messages frequency.

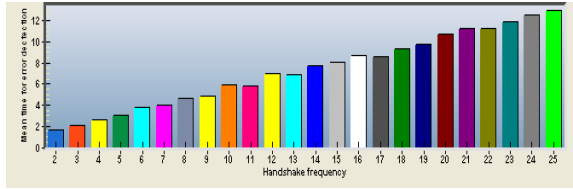
With the second test (Figures 2 and 3), we can see that the required time to create new communication links between agents depends on the number of tasks per agent. In fact, the more tasks there are, the more replica per agent may exist, and then the more links between agents must be established since



**Fig. 4.** Rate of false failure detection with 40 agents



**Fig. 5.** Rate of false failure detection with 200 agents



**Fig. 6.** Mean time for fault detection by handshake frequency

the agents that previously were communicating with the failed agent have to communicate with new agents in which the failed tasks are replicated.

For the third test (Figures 4 and 5), we can see that the rate of false failure detection decreases when the number of agents is high. In fact, the more agents there are in the MAS, the more links between agents are established, and the more replica per agent there are. Consequently, the probability that all the handshake messages with an agent  $a_i$  are lost decreases when the number of  $a_i$ 's replica increases. We can see that there is approximately no false detection with a 200 agents system (see Figure 5) with a loss messages rate of 60%. The more agents the system has and the less false failure detections are done.

From the fourth test (see Figure 6), we can see that the mean time to detect a failure in a MAS is approximately half the time of the frequency of handshake messages. For example, if the handshake messages frequency is 25 seconds, then it will take, at mean time, about 12 seconds for the MAS to detect an agent failure if it occurs. The more the MAS is critical, the more we must detect failures in time, and the more we have to use short time handshakes.

From these tests, first, we conclude that our technique is scalable since it can be applied to 200 agents but it is limited to 300 agents. At 300 agents, our program is out of memory. The technique must be improved to deal with more



than 300 agents. Second, the duration time taken to establish new communication links depends on the number of agents and on the number of tasks per agent; it is about 14 seconds with a system having 200 agents and 2 tasks per agent. Third, the mean time taken to detect an agent failure is about half of the handshake frequency. Consequently, the more critical the system is, the shorter the handshake frequency should be.

Our reorganization strategy does not guarantee that the MAS will properly operate. It should be combined with other techniques which deal with programming failures and exceptions such as those described in [7], [8], [10], or [11]. Our reorganization strategy only addresses the failures which come from the environment.

## 5 Discussion

There are several works done for building fault-tolerant multi-agent systems such as those based on exceptions handling [10] or on task delegation [7]. However, we cannot compare ourselves to these works since we are not addressing the same issue because first, we are not dealing with exceptions handling on multi-agent systems. As stated earlier, we are focusing on errors originating from the MAS environment and not from programming exceptions. Second, fault-tolerant techniques based on task delegation are based for example on a trust model as described in [7] and which determines whether the risk for an agent partner to fail is minim. However, such techniques do not address the issue when the task in execution is in failure and this is what our strategy tries to address. Hence, we compare our work to the techniques based on agents replication.

Before presenting this comparison, we briefly describe two relevant fault-tolerant techniques based on agent replication. The first technique is implemented in the *DARX* framework [8] used to develop fault-tolerant multi-agent systems. It is based on data and/or computation replication. *DARX* allows to automatically and dynamically apply replication mechanisms to agents. The second technique helps brokers recover from failures [11]. It is based on broker replication. It is applied when there are several broker agents in a multi-agent system [11]. These broker agents may be able to substitute for any broker agent that becomes unavailable. Hence, the multi-agent system can continue to operate as long as there is at least one broker agent remaining idle. The agents, that were communicating with the failed broker will subscribe to this new broker, and communication will resume.

The replication strategy we propose is different from the replication strategies proposed in the literature [8] [11]. Our strategy is based on task and agent replication and not on only agent replication as in [8] [11]. In our evaluation, we were able to demonstrate the impact that the number of tasks per agent has on the performance of the system (see Figures 3 and 4).

Our strategy is well suited to critical systems in which high performance is required. In fact, our strategy determines the MAS' environment as a boundary between what the MAS controls and what it does not control. Consequently new

sources of failures related to the outside environment can be identified from the start, during the design phase that the techniques described in [8] and [11] do not allow. Hence, several recovery plans may be anticipated and added to agents action plans in order to overcome unexpected failures originating from the environment. This should guarantee more fault-tolerance to the system since new sources of failures are taken into account in the development phase of the MAS.

## 6 Conclusion

In this paper, we defined a reorganization strategy which proposes a failure categorization that it has not been proposed in the literature. It allows determining the conditions under which the MAS may recover from its failures and those from which it may not. The reorganization strategy includes a fault-tolerance and a fault-recovery techniques. The fault-tolerance technique determines the critical agents of the system which must be replicated. The other agents of the system will only see their tasks being replicated in other agents. Hence, this technique decreases the resulting system complexity by minimizing the number of agents to replicate. Furthermore, if it is costly to replace an agent by another one, then this technique allows to minimize such a cost. The fault-recovery technique allows the agents to detect and overcome failures originating from the MAS' environment.

As future works, first, we have to improve the reorganization strategy by increasing the number of agents on which its can be performed (more than 300). Second, this strategy can be adapted to deal with different application domains such as supply chain management problems. For example, let us consider a supply chain with several warehouses. If a disaster occurs, the supply chain must continue operating. If each warehouse is considered as an agent, then the sources of disasters describe the environment of the supply chain. Since it is not possible to replicate all the warehouses, the technique can be applied and adapted to deal with such problems. Finally, this strategy must be integrated within a more general framework such as a MAS design methodology in order to support MAS designers when building fault-tolerant multi-agent systems.

## References

1. Bolchini, D., and Mylopoulos, J., (2003) From Task-Oriented to Goal-Oriented Web Requirements Analysis, Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE03), pp. 166-175.
2. Bora, S., and Dikinelli, O. (2006) 'Implementing a Multi-agent Organization that Changes Its Fault tolerance Policy at Run-Time', In O. Dikinelli, M-P. Gleizes, and a. Ricci (eds): *ESAW 2005, LNAI 3963*, pp. 153-167.
3. Campelo, J.C., Rodriguez, F., and Serrano, J.J. (1999) 'Dependability evaluation of fault tolerant distributed industrial control systems', In *Workshop on Parallel and Distributed Real-Time Systems*, pp 384-388.
4. Cragg, L., Tsui, P.W., and Hu, H. (2003) 'Building a fault tolerant architecture for internet robots using mobile agents', In *Proceedings of 1st Workshop on Internet and Online Robots*.

5. Farooq H., A., and al. (2005) 'Scalable Fault tolerant Agent Grooming Environment - SAGE', *In Proceedings of the fourth international conference on autonomous agents and multi-agent systems (AAMAS'05)*.
6. Fedoruk, A., and Deters, R. (2002) 'Improving fault-tolerance by replicating agents', *In Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 737-744.
7. Griffiths, N. (2005) 'Task delegation using experience-based multi-dimensional trust'. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, The Netherlands, pp. 489-496.
8. Guessoum, Z., Briot, J.P., and Charpentier, S. (2002) 'A fault-tolerant multi-agent framework', *In Proceedings of the First International Joint Conference on Autonomous agents and multiagent systems*, pp. 672-673.
9. Hagg, S. (1996) 'A sentinel approach to fault handling in multi-agent systems', *In Proceedings of the Second Australian Workshop on Distributed AI, in conjunction with the Fourth Pacific Rim International Conference on Artificial Intelligence (PRICAI'96)*.
10. Klein, M., RODRIGUEZ-AGUILAR, J.A., and DELLAROCAS, C.(2003) 'Using Domain-Independent Exception Handling Services to Enable Robust Open Multi-Agent Systems: The Case of Agent Death'. *Journal for Autonomous Agents and Multi-Agent Systems*. Volume 7, Number 1/2.
11. Kumar, S., and Cohen, P.R. (2000) 'Towards a fault-tolerant multi-agent system architecture', *In Proceedings of the fourth international conference on Autonomous agents*, pp. 459-466.
12. Luck, M., Ashri, R., and D'inverno, M. (2004) 'Agent-based software development', *In Artech House Eds*, 208 pages, ISBN: 1-58053-605-0.
13. Pullum, L.L. (2001) 'Software fault tolerance techniques and implementation', *In Artech House eds*, 360 pages, ISBN: 1580531377.

# A Multi-site Subcellular Localizer for Fungal Proteins

Michel Nathan

Department of Computer Science & Software Engineering  
Concordia University

**Abstract.** In this work, we build a decision tree to predict fungal protein localization based on physiochemical properties of proteins calculable from their primary sequences. Although there is clear evidence of presence of the same protein in more than one sub-cellular compartment, almost all existing automated systems restrict their predictions to single-site localization. We address this issue by predicting as many localization sites as possible. When localizing among 17 sub-cellular compartments, in 64% of the cases our system successfully predicts at least one of the experimentally reported localizations. Moreover, all reported localizations are correctly predicted in 49% of the cases. We also report 76 fungal protein features expected to be implicated in localization, based on the constructed decision tree.

**Keywords:** subcellular localization, fungal protein, decision tree, multi-site.

## 1 Introduction

Sub-cellular localization allows biologists to make inference on the functions of a protein and its annotation. It also provides important hints on the pathways the protein is involved in and the class of proteins it may interact with. Biochemical, cytological and genetic methods are used for functional characterization of known proteins as well as search of new proteins. The results of such experiments provide accurate and reliable information on sub-cellular compartments that are targeted by proteins. These experiments are, however, labour intensive and manual annotation cannot keep up with the increasing number of gene products that become available. Automated predictors have been built in recent years in an effort to accelerate the localization process. These systems use classification schemes that are based on one or more of the following approaches: i) Ab-initio methods that use compositional, bio-chemical or structural features of proteins to predict their localization (ex: PLOC [1]), ii) Methods based on sorting signals that determine a protein's target location (ex: TargetP [2]), (iii) Signature-based methods that use motifs and profiles to characterize protein families and family domains that, in turn, serve in localization (ex: PSLT [3]), and (iv) Homology-based methods (ex: Proteome Analyst [4]). There are also predictors that are based on a combination of the 4 mentioned methods. The best known of such

hybrid systems is PSORT [5] and its extensions (PSORT II, PSORTb, iPSORT and WoLF PSORT) that use a knowledge-based system to predict localization based on a protein's overall amino acid composition, targeting signals and motifs.

It is not known a priori which specific features or combination of features of a protein play determining roles in its localization. Therefore, all biochemical and environmental factors are potentially representative and should undergo scrutiny. Moreover, it is widely recognized that proteins localize to more than one sub-cellular site [10]. A predictor should therefore also be able to handle multi-site localization. We propose to build a classifier that learns, from experimentally derived localization information, which bio-chemically meaningful features of proteins are relevant to their localization. The system then uses this knowledge to determine, given an amino acid sequence of a protein, all sub-cellular locations to which the protein is destined. Two important aspects of our system need to be emphasized: (i) the use of a hybrid classification approach based on bio-chemical properties of proteins captured through their amino acid compositions, targeting signals as well as functional motifs, and (ii) multiple-site localization, i.e., the capacity to determine all the destinations within the cell to which a given protein may be targeted. Fungi were chosen as organism group for our investigation because they are endowed with numerous organelles and sub-cellular locations. Moreover, extensive experimental data related to fungi may be found in the literature.

## 2 Methods and Materials

### 2.1 Methodology

We obtain a significant number of experimental examples localizing various fungal proteins in numerous compartments of the cell. For each element of the set thus obtained, i.e. for each protein of known sub-cellular localization, we compute a set of pre-determined characteristic features purely based on the information contained in its amino acid sequence. Using the reported localizations and the feature values calculated from the collected examples, we build a feature-location matrix that associates each set of feature values with its corresponding localization. This matrix is input into a learning tool (ID3) that generates a decision tree that classifies the proteins according to their targeted locations. By analyzing this decision tree, we shall attempt to find out which of the selected features have no or little impact on the targeting of proteins to a given location, and which sub-cellular locations cannot be localized given our set of training examples.

### 2.2 Data Source

For the purpose of the current work, we selected 17 localization sites (Table-1) from the Cellular Component portion of Gene Ontology (GO). The criteria used to guide the selection were: (i) Biological importance of the compartment - for example nucleolus is considered as a distinct site of localization, and (ii) Availability of evidences of experimentally discovered localizations. We also downloaded

**Table 1.** Count of experimentally reported localizations per site

Cellular Site	Count of Reported Localizations
Cell Cortex	72
Cell Wall	140
Cytoplasm	2821
Endosome	39
Endoplasmic Reticulum (ER)	74
Extracellular Region	35
Golgi	50
Mitochondrial Inner Membrane	73
Mitochondrial Matrix	42
Mitochondrial Outer Membrane	23
Mitochondrial Intermembrane Space	26
Nuclear Envelope	81
Nucleus	1755
Nucleolus	147
Peroxisome	24
Plasma Membrane	211
Vacuole	46
Total	5659

localization data for *Saccharomyces cerevisiae*, *Candida albicans* and *Schizosaccharomyces pombe* from annotation section of GO (Release: 13 Feb. 2006) [6]. The current experimental finding of fungal protein localizations is far from complete and most of the localizations found in the selected sources (as well as elsewhere in the literature) are inferred from electronic annotation or from sequence or structural similarity. In order to choose more biologically significant localizations, from the downloaded data we solely selected the experimentally evidenced localizations, i.e. those inferred from direct assay, genetic interaction, mutant phenotype or physical interaction. We thus compiled, as our data set, 5659 reported localizations for the 17 sites of Table-1. The complete listing of 4529 proteins in our data set may be found in Table-A5 of [7]. Among these proteins, 3458 are reported to localize to a single site and the remaining ones are multi-site (1014, 55 and 2 proteins are reported to localize to 2, 3 and 4 sites respectively).

### 2.3 Selection and Calculation of Protein Features

Eukaryotic cells are endowed with many sub-cellular organelles and non-organelle structures. Each of these structures exists within a unique biochemical environment and performs one or more specific functions. Three major aspects of proteins potentially related to their localization are: i) physiochemical properties, ii) molecular functions, and iii) biological pathways. We propose to consider three categories of protein features, calculable from their primary sequence, as characteristics associated with the three mentioned aspects. These are compositional features, functional motifs, and targeting motifs, respectively.

*Compositional features.* Length, molecular weight, and isoelectric point are three important physical features of proteins. The latter two features are often used in protein separation methods. As for chemical features, it has been shown that there is a correlation between amino acid composition and cellular location of proteins [8]. We propose to consider the following distinctive chemical features: a) amino acids percentage composition, b) distribution of residue size (tiny, small, medium-sized and bulky), c) basic, acidic, uncharged-polar, non-polar, aromatic and aliphatic composition, d) hydrophobicity (hydrophobic, weakly-hydrophobic and hydrophilic content), and e) dipeptide composition. In total, 42 physiochemical features were selected for this category, a list of which is found in Table-A2 of [7].

*Functional motifs.* Most of the proteins can be grouped, on the basis of similarities in their structure and function, into a limited number of families and family domains. PROSITE database [9] is a compilation of protein families. It uses one or more patterns (functional motifs) to represent each protein family. Through interaction with environmental factors within the cell, these functional motifs confer to the proteins the specific functionality that characterizes the family. We propose to extract these functional motifs from PROSITE and use them as characteristic features for fungal protein localization. On release 19 of PROSITE, we found 119 patterns (regular expressions) associated with fungal proteins. These patterns are listed in Table-A3 of [7] and we study them as potential factors implicated in localization to specific sites.

*Targeting motifs.* Proteins that take part in a given biological process are often from diverse families. Therefore, family-specific functional motifs alone are not sufficient for targeting proteins in the pathways in which they are involved. The final address of proteins that enter the secretory pathway is largely specified by short signals (motifs) within the protein that determine interaction with particular elements of the pathway [10]. Indeed, numerous experimental studies provide well-documented information on how the sorting receptors within the cell recognize various motifs in proteins amino acids sequences and selectively target them to appropriate destinations. Through review of findings in the literature, we selected a set of 17 motifs (Table-2) that have been recurrently associated with targeting of specific cellular compartments. Further details on the selected motifs, that are also regular expressions, may be found in Table-4 of [7].

Considering the three mentioned categories, a total number of 178 features were selected for localization study. For each example protein in our data set, we calculated each of the compositional feature values as well as Boolean values indicating presence/absence of each of the mentioned functional and targeting motifs. We thus constructed a feature-localization matrix with each tuple consisting of a protein name, a reported localization site and values of the 178 selected features. For each protein that is reported to localize to multiple sites, this matrix includes as many tuples as there are reported localizations.

**Table 2.** Protein targeting motif features and their corresponding hypothesized target sites

Feature name	Hypothesized Targeting Site
Cleavable signal Peptide (CLSP)	Extracellular Region
Transmembrane segment (TMS), Mature protein Transmembrane Segment (MTMS), Length of MTMS (MTMSLEN)	Membrane
ER retention/retrieval signal (ERRS)	ER
Endosomal signal (ES)	Endosome
Vacuolar targeting signal (VTS)	Vacuole
GPI attaching signal (GAS)	Cell Wall
ER transmembrane segment (ERTMS)	ER membrane
Vacuolar transmembrane segment (VTMS)	Vacuole
Nuclear membrane localization signal (NMLS)	Nuclear Envelope
Nuclear localization signal (NLS)	Nucleus
Peroxisomal targeting signal (PTS)	Peroxisome
Peroxisomal membrane signal (PMS)	Peroxisomal membrane
Mitochondrial transfer peptide (MTP)	Mitochondrial inner/outer membrane
Mitochondrial matrix transport signal (MMTP)	Mitochondrial Matrix
Vesicular signal (VS)	Cytoplasmic membrane-bound Vesicles

## 2.4 Localization Based on Decision Tree

A decision tree is built by applying ID3 [11] on our feature-localization matrix. Classification of a query protein is accomplished by starting at the root of the decision tree and traversing the tree until a terminal node is reached. The localization site associated with this terminal node then represents the prediction for our query protein. During the tree traversal, the branch taken at each node is determined by the value of the query protein for the feature associated with that node. If a node is reached at which no emanating branch corresponds to the feature value of the query protein, then no prediction could be made for this query protein. If, on the other hand, the tree traversal leads to a certain node at which no decision could be made as to which branch to take next (this could occur, for example, when the set of examples associated with that node consists of 2 or more proteins with identical feature values but different localization sites), then we consider such a node as a multi-site terminal node and predict the query protein to target all the localizations associated with that node.

## 3 Results

### 3.1 Performance

To evaluate the performance of the system, we followed a 5-fold cross-validation. We divided our data set into five subsets of approximately equal size ensuring that: a) single-site, double-site and triple-site proteins are each equally distributed among the five subsets, b) quadruple-site proteins (two in our data set) are distributed into distinct subsets, and c) each subset contains approximately the same proportion of examples located to each sub-cellular site. In each of the five cross validations, a distinct set among these five subsets was used as



**Table 3.** Per protein performance evaluation measures for localizations

Localization	Protein Count	Partial Prediction	Total Prediction	Identical Prediction
Single-Site Proteins	3458	58%	58%	30%
Double-Site Proteins	1014	82%	20%	10%
Triple-Site Proteins	55	84%	5%	0%
Quadruple-Site Proteins	2	100%	0%	0%
All Proteins	4529	64%	49%	25%

the query set and the totality of the remaining 4 subsets was used as training set. The set of experimentally reported localizations was then compared to the corresponding prediction set using the following performance criteria: i) Partial prediction: system correctly predicts some of the reported localizations; we determine the percentage of proteins for which at least one true positive (TP) prediction was made, ii) Total prediction: system correctly predicts all the reported localizations but may also predict some that are not reported; we determine the percentage of proteins for which no false negative (FN) prediction was made, iii) Identical prediction: system correctly predicts all the reported localizations and none other; we determine the percentage of proteins for which neither FN nor false positive (FP) prediction was made. Table-3 depicts the results. There were 76 proteins for which no localization could be predicted by the system. These are listed in Table-9 of [7].

To measure the effect of the variability of the number of training examples on the system’s predicting power, we calculated the proportion of correct prediction of single-site proteins for each of the designated localization sites. The result (Table-4) clearly shows a correlation between the number of training examples and the proportion of correct localization. In the lower extreme, no correct prediction could be made for the mitochondrial matrix and outer membrane as well as the peroxisome. As a general finding, all the proteins (single-site or multiple-site) for which the system could not correctly predict any of the reported localizations had 25 or less examples in the training set.

### 3.2 Characteristic Features

In order to determine the features that are effective in localization, a full decision tree was built for 5659 examples using 178 features. We found that 76 feature values were actually used in the decision tree in order to determine the localizations. These features are therefore expected to play a role in sorting the target locations. Table-5 provides a list of these features.

### 3.3 Targeting Motifs Validation

The constructed decision tree is used to validate the hypothesized targeting motifs. Targeting motif feature values attached to each node of the decision tree built from our data set are compared to values proposed to favour specific targets as per Table-2. Table-6 shows the frequency of occurrence of the targeting motifs in proteins that localize to 12 corresponding sub-cellular sites. From the

**Table 4.** Proportion of correct localization as a function of number of training examples

Reported Localization Site	Examples	Correct Prediction	Percent Correct Prediction
Cytoplasm	1878	1290	69%
Nucleus	981	551	56%
Cell Wall	101	46	46%
Plasma Membrane	133	59	44%
Extracellular Region	23	9	39%
Golgi Apparatus	20	5	25%
Nucleolus	51	12	24%
Mitochondrial Inter-membrane Space	11	2	18%
Mitochondrial Inner Membrane	48	8	17%
Vacuole	25	4	16%
Nuclear Envelope	33	4	12%
Endoplasmic Reticulum	38	4	11%
Cell Cortex	49	4	8%
Endosome	20	1	5%
Mitochondrial Matrix	25	0	0%
Mitochondrial Outer Membrane	15	0	0%
Peroxisome	7	0	0%

result in the last six rows (indicating that only less than or equal to 38% of the analyzed proteins contain the proposed targeting motifs), even after considering the margin of error, we can claim that the stated targeting motifs are not sufficient for localization of fungal proteins to peroxisome, vacuole, nuclear envelope, mitochondrial matrix, endosome and cell wall.

## 4 Discussion

### 4.1 Containment Factor

Lack of sufficient protein examples localized to one or more sub-cellular sites is the main obstacle in predicting correctly the localizations, as shown in Table-4. Another hindrance may be the containment factor resulting from the use of "part-of" relationship when we compiled our data based on GO classification. For example a protein reported to localize to the nucleolus but predicted to go to the nucleus or one reported to target the cell cortex but predicted to go to the cytoplasm are each counted as a false positive (FP) and a false negative (FN). On the other hand, the fact that the system predicts, quite correctly, that localization occurs in the nucleus and cytoplasm (respectively) is not considered as true positive (TP). This containment factor is a result of the hierarchical nature of the cellular compartments. To measure the extent of impact of such containment on system's prediction, we re-calculated the system performance as we did before (Table-3) but this time we considered a given predicted localization site as true positive (TP) when it is identical to or when it contains the experimentally reported localization site (ex: when a localization is reported to be nucleolus, prediction of nucleolus and nucleus are both considered as TP). Performance evaluation measures for this trial were 67%, 54% and 35% for partial, total and identical predictions respectively. Comparison of these results with those of Table-3 indicates that containment factor has some effect on the correctness of the localizations.

**Table 5.** 76 Characteristic fungal protein features implicated in sub-cellular localization

Feature Type	Feature Name
Physical properties	Length, molecular weight, isoelectric point
Chemical properties	Acidic, basic, aliphatic, aromatic, polar and uncharged, non-polar
Size distribution	Tiny, small, medium, bulky
Hydrophobicity	Hydrophobic, Weakly hydrophobic, Hydrophilic, Hydrophobicity
Dipeptide composition	LL, LS, SL
Amino acid composition	A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y
Functional Motifs	Malate dehydrogenase active site signature
	Multicopper oxidases signature 1
	Malic enzymes signature
	Actins signature 1
	Profilin signature
	Glutamine amidotransferases class-I active site
	FGGY family of carbohydrate kinases signature 2
	Ribosomal protein S5 signature
	Glycosyl hydrolases family 10 active site
	Indole-3-glycerol phosphate synthase signature
	Glycosyl hydrolases family 2 signature 1
	Glycosyl hydrolases family 11 active site signature 1
	Glycosyl hydrolases family 11 active site signature 2
	Chorismate synthase signature 3
	EPSP synthase signature 2
	Cutinase, aspartate and histidine active sites
	Glyoxalase I signature 1
	Imidazoleglycerol-phosphate dehydratase signature 2
	Dehydroquinase class I active site
	DNA/RNA non-specific endonucleases active site
	Chitinases family 18 active site
	Casein kinase II regulatory subunit signature
	N-acetyl-gamma-glutamyl-phosphate reductase active site
	Cleavable signal Peptide
	ER retention / retrieval signal
	ER transmembrane segment
	Endosomal signal
	GPI attaching signal
	Mitochondrial transfer peptide
	Nuclear localization signal
	Nuclear membrane localization signal
	Peroxisomal membrane signal
	Peroxisomal targeting signal
	Vesicular signal
	Vacuolar targeting signal
	Transmembrane segment
Targeting Motifs	

4.2 Performance Comparison with Existing Localizers

The State of the Art localization predictor, Proteome Analyst [4], achieves a precision of 87% when set to predict localization of fungal proteins among 9 sub-cellular sites. We note, however, that Proteome Analyst’s classifier selects the most probable localization and does not report multiple-site localizations. Similarly, many other existing predictors do not consider this multiplicity and predict only the best possible localization. Among the systems that do output more than one prediction for a given protein is the PSORT [5] family of localizers. These localizers provide as output, the probabilities of targeting to individual sub-cellular compartments. WoLF PSORT, the most recent member of the PSORT family, specializes in animal, plant and fungal proteins. An overall prediction accuracy of 83% has been reported for a 14 compartment

**Table 6.** Occurrence of targeting motifs in proteins of the targeting set

Localization Site	Protein	Targeting Motifs	Count Protein with motifs	Percent Localized to proposed site
Mitochondrial inner membrane	73	MTP	73	100%
Mitochondrial outer membrane	23	MTP	23	100%
Nucleus	1755	NMLS / NLS	1663	95%
ER	74	ERRS / ERTMS	62	84%
Plasma membrane	211	TMS / MTMS	170	81%
Extracellular Region	35	CLSP	25	71%
Peroxisome	24	PTS / PMS	9	38%
Vacuole	46	VTS / VTMS	9	20%
Nuclear Envelope	81	NMLS	5	6%
Mitochondrial Matrix	42	MMTP	0	0%
Endosome	39	ES	0	0%
Cell Wall	140	GAS	0	0%

localization of yeast by WoLF PSORT [12]. These 14 compartments consist of 10 single-site locations and 4 dual localization sites (cytoplasm and nucleus, cytoplasm and mitochondrion, cytoplasm and peroxisome, and finally mitochondrion and nucleus). WoLF PSORT analyzes the sequence using pre-determined rules having as pre-conditions information on the type of targeting signal motifs that direct proteins to various sub-cellular locations. WoLF PSORT also uses some correlative sequence features such as amino acid content from iPSORT [13] and sequence length. Although the latter features contain non-causal ab-initio information about the sorting signals, causal components of localization reflected through rules continue to drive the decision process.

There is a risk involved in attempting to guide the localization process through rules and pre-conditions as many of such rules are incomplete. In fact, as it was shown in Table-6, in many cases the presence of a targeting motif is not a sufficient cause for localization to a particular site. Moreover, there are ambiguities involved in determining the exact pre-condition to which some of these targeting motifs should correspond. For example, a trans-membrane helix is indicative of the presence of a targeting motif that favours localization to the plasma membrane or to the membrane of an organelle. On the other hand, the presence of a secretory signal peptide implies direction of the protein to the extracellular region. Now, it is hard to distinguish between these two signals. In fact, it has been shown that the presence of one signal in a protein consistently interferes with the detection of the other [14].

The only existing single-species predictor that does consider multi-site localization in a rigorous manner is the one devised by Chou and Cai [15]. This system uses a data set of 3875 proteins that are experimentally localized into one or more of 22 sub-cellular locations in budding yeast [16]. This amounts to a total of 5132 reported localizations or  $(p,s)$  pairs where  $p$  stands for a protein and  $s$  for a single sub-cellular site. To define a protein's feature space, this system uses a dimensional vector of 9772 entries consisting of 3 categories of information: a) InterPro-mapped-to-GO, b) functional domain composition, and c) pseudo amino acid composition. For a given query protein,  $p$ , its feature vector,  $v$ , is compared to the set of feature vectors,  $V$ , of 5132 training proteins. The reported

localization(s) of those elements of  $V$  that show the highest similarity with  $v$  are designated as the system prediction(s) for the query protein. For example, if the three vector features  $(p, s_1)$ ,  $(p, s_2)$  and  $(p, s_3)$  depict the highest similarity with  $v$  then  $p$  is predicted to localize to the set consisting of the union of  $s_1$ ,  $s_2$  and  $s_3$ . The performance measure reported by Chou and Cai [15] consists of the standard per site sensitivity measure  $TP / (TP + FN)$ . The reported values are 70%, 84% and 90% (respectively) when the highest-ranking, the two highest-ranking, and the three highest-ranking predictions (respectively) in terms of similarity with experimental results are taken into consideration. No mention of FP and the impact of considering more than one highest-ranking prediction on the number of FP has been made in their work.

For the purpose of comparing our system's performance with that of Chou and Cai, we also proceeded with per-site evaluation of sensitivity of our system. For each protein, we counted the number of sites that were correctly predicted, summed this number over all the proteins and divided the result by the total number of sites reported. We thus obtained a per-site sensitivity of 55%. Chou and Cai did not, however, report the number of false positive (FP) and how this number increases as they would consider more than one (2 or 3) top ranking candidates for the purpose of localization based on similarity. As a performance measure that reflects FP we also calculated the per-site specificity of our system and obtained 91%. Furthermore, it is important to note that Chou and Cai's system deals solely with budding yeast and does not perform multi-species prediction as is attempted in this work.

### 4.3 Design Decisions

Studies of dipeptide frequencies in the literature reveal that GF, GY, NG, NT are among the dipeptides that occur most frequently in proteins in general [17]. However, to our knowledge, no results have been reported for this occurrence specifically in fungi. We use the amino acid sequences of a set of representative fungi to find the dipeptides that predominantly occur in fungal proteins. Calculations performed on a large set (120734) of fungal proteins from 20 different fungal species from RefSeq database of NCBI ([http://www.ncbi.nlm.nih.gov/RefSeq/Release 7](http://www.ncbi.nlm.nih.gov/RefSeq/Release%207)) revealed that 5 specific dipeptides (SS, LL, LS, AA, SL) occur more often (20% or more frequent) than others in fungal protein sequences. We select these dipeptides as features to study for localization.

We chose functional motifs (as represented by patterns in PROSITE) as protein signature for determination of localization. This choice was motivated by the following considerations: i) a motif is usually located in a short well-conserved region — typically an enzyme catalytic site, prosthetic group and an attachment site [18], ii) motifs have a higher frequency of occurrence than domains, and iii) a motif is a regular expression that is a quantitative descriptor: it either matches or it does not.

Our choice of ID3 as the learning tool was motivated by the following considerations: i) Our training examples are known at the time of building the classifier so there is no need for incremental learning, ii) ID3 uses all training examples at each step in the search to make statistically based decisions regarding how

to refine its current hypothesis and so the resulting search is much less sensitive to errors in individual training examples, and iii) ID3's inductive bias has been proven to provide a high efficiency for very large sizes of training set [11].

#### 4.4 System Coverage

Three types of coverage need to be considered in evaluating a sub-cellular localization problem: i) Location coverage: sub-regions that are supported by a predictor. In this work, we selected 17 sub-cellular compartments as representative of all cellular components found in GO (Table-1); ii) Sequence coverage: ratio of sequences for which a prediction is made to the total number of sequences of interest. The total number of query proteins for which the system could not predict any localization was 76. Out of a total of 4529 query proteins, this amounts to  $(76/4529 =) 1.7\%$  of the total set considered, hence a sequence coverage of 98.3%; (iii) Taxonomic coverage: measures the range of organisms that the predictor covers. In our work, we have considered 3 species in fungi kingdom for which we have found extensive experimental localization information.

#### 4.5 Hierarchy and Multiple Classification Issues

Gene ontology cellular components form a directed graph and not a tree. For example, both *cell cortex* and *site of polarized growth* have *polarisome* as a direct descendant and a protein reported to localize to the latter site may be classified as belonging to either of the two former sites. Moreover, some prominent sub-cellular compartments are fully contained within other compartments. *Nucleolus* found within *nucleus* and *cell cortex* contained within *cytoplasm* are two such examples. In many cases there is no apparent solution to this hierarchy issue. For example, the nuclear proteins that are not localized to the nucleolus or to the nuclear envelope are not reported as such. Instead, they are rather simply identified as belonging to the nucleus. All these reported localizations are nonetheless significant and need to be equally considered as representative examples. But then, how can a predictor distinguish between such classes? A predictor that is seeking differences in protein features that would explain distinct targeting could not utilize such examples. So the information contained in such examples hampers classification of proteins into a single category. These considerations necessitate multiple localizations of the same protein to different locations, the problem that we address in this work.

### 5 Conclusion

Proteins target those sub-cellular locations where they can perform their functions. They are guided to their respective destinations through interaction with other proteins (Protein-Protein interaction, PPI). Specificity of such interactions is presumed to be due to physiochemical and biological characteristics inherent in the very protein considered. Molecular functions of a protein as well as the biological processes in which it takes part are also expected to influence the choice

of its target destination. Specificity of such aspects need to be characterized by appropriate protein family signatures as well as targeting sequence signals that allow recognition of the proteins by specific receptors of a given process.

In this work, we have started with a set of features that can potentially impact PPI, protein function and the biological processes that the protein is involved in. A uniform, non-causal method has then been deployed to determine the features that truly correlate with any of the sub-cellular localizations. Our developed algorithm is non-causal in the sense that no pre-established rule has been introduced that would specifically favour the localization of proteins with certain motifs to particular compartments. In spite of our simple approach, using a classical machine learning method, and our restriction of the feature space to utilize solely the knowledge elicitable from the protein's primary sequence, the developed system succeeded in correctly predicting 64% of the multiply-localized proteins. This result is indicative of our distinctive strategy that uses the decision tree and an ab-initio approach to handle the problem of sub-cellular localization. We have also contributed towards elucidation of protein characteristics that correlate with localization to sub-cellular sites by identifying and reporting 76 features that are used in our decision tree. Lastly, in contrast to other multi-site localizers built and evaluated based on a single organism (yeast), in this work we have attempted to take advantage of the variability of the sorting mechanisms that may be found in different species. This work is a first attempt in multi-site, multi-species localization of fungal proteins.

The system could not predict the localizations of proteins to the mitochondrial matrix, the mitochondrial outer membrane and the peroxisome due to the lack of sufficient training examples. There is ongoing work to collect more examples as they become available in order to enlarge our dataset. Experimental localization results for organisms other than the 3 species considered here may also be used. In upcoming work, we also intend to improve the system's performance by expanding the feature set to include structural information on proteins. Furthermore, we shall investigate the distinguishing features of proteins that account for localization to specific sites. In particular, we shall attempt to identify the features that are necessary for localization to each site.

**Acknowledgements.** This work was done as part of a Masters thesis in Computer Science Department of Concordia University under the supervision of Professor Greg Butler.

## References

1. Park, K. J., Kanehisa, M. : Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. *Bioinformatics* **19**, 1656-1663 (2003)
2. Emanuelsson, O., Henrik Nielsen, H., Brunak, S., von Heijne, G. : Predicting sub-cellular localization of proteins based on their N-terminal amino acid sequence. *J. Mol. Biol.* **300**, 1005-1016 (2000)

3. Scott, M. S., Thomas, D. Y., Hallett, M. T. : Predicting Subcellular Localization via Protein Motif Co-Occurrence. *Genome Research* **14**, 1957-1966 (2004)
4. Lu, Z., Szafron, D., Greiner, R., Lu, P., Wishart, D. S., Poulin, B., Anvik, J., Macdonell, C., Eisner, R : Predicting subcellular localization of proteins using machine-learned classifiers. *Bioinformatics* **20**(4), 547-556 (2004)
5. Nakai, K., Kanehisa, M. : A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics* **14**, 897-911 (1992)
6. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G. : Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genet.* **25**(1), 25 - 29 (2000)
7. Nathan M. : A Multiple Site Predictor for Subcellular Localization Of Fungal Proteins. Masters Thesis. Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada (2006)
8. Cedano, J., Aloy P., Perez-Pons, J. A ., Querol, E. : Relation between amino acid composition and cellular location of proteins. *JMB*, **266**(3), 594-600 (1997)
9. Hulo N., Bairoch A., Bulliard V., Cerutti L., De Castro E., Langendijk-Genevaux P.S., Pagni M., Sigrist C.J.A : The PROSITE database. *Nucleic Acids Res.* **34**, D227-D230 (2006)
10. Pringle, J. R., Broach, J. R., Jones, E. W. : The Molecular and Cellular Biology of the Yeast *Saccharomyces*. Cold Spring Harbor Laboratory Press, USA (1997)
11. Quinlan, J. R. : Learning efficient classification procedures and their application to chess end games. In: Editors, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell : *Machine Learning - An Artificial Intelligence Approach*, 463-482. Tioga, Palo Alto, CA, USA (1983)
12. Horton, P., Park, K. J., Obayashi, T., Nakai, K. : Protein Subcellular Localization Prediction with WoLF PSORT. In: *Proceedings of the 4th Annual Asia Pacific Bioinformatics Conference APBC06*, Taipei, Taiwan. 39-48 (2006)
13. Bannai, H., Tamada, Y., Maruyama, O., Nakai, K., Miyano, S. : Extensive feature detection of N-terminal protein sorting signals. *Bioinformatics* **18**, 298-305 (2002)
14. Lao, D. M., Okuno, T., Shimizu, T. : Evaluating transmembrane topology prediction methods for the effect of signal peptide in topology prediction. In *Silico Biology* **2**, 485-494 (2002)
15. Chou, K. C., Cai, Y. D. : Predicting protein localization in budding Yeast. *Bioinformatics* **21**(7), 944 - 950 (2005)
16. Huh, W. K., Falvo, J. V., Gerke, L. C., Carroll, A. S., Howson, R. W., Weissman, J. S., O'Shea, E. K. : Global analysis of protein localization in budding yeast. *Nature* **425**, 686-691 (2003)
17. Champion, S. R., Ameen, A. S., Lai, L., King, J. M., Munzenmaier, T. N. : Dipeptide frequency/bias analysis identifies conserved sites of nonrandomness shared by cysteine-rich motifs. *Proteins* **44**, 321-328 (2001)
18. Sigrist C.J.A., Cerutti,L., Hulo,N., Gattiker,A., Falquet,L., Pagni,M., Bairoch,A. and Bucher,P. : PROSITE: a documented database using patterns and profiles as motif descriptors. *Brief. Bioinform.* **3**, 265-274 (2002)



# Selecting Genotyping Oligo Probes Via Logical Analysis of Data<sup>\*,\*\*</sup>

Kwangsoo Kim and Hong Seo Ryoo<sup>\*\*\*</sup>

Division of Information Management Engineering, Korea University  
1, 5-Ka, Anam-Dong, Seongbuk-Ku, Seoul, 136-713, Korea  
Phone: +82-2-3290-3394, Fax: +82-2-929-5888  
{kks00,hsryoo}@korea.ac.kr

**Abstract.** Based on the general framework of logical analysis of data, we develop a probe design method for selecting short oligo probes for genotyping applications in this paper. When extensively tested on genomic sequences downloaded from the Los Alamos National Laboratory and the National Center of Biotechnology Information websites in various monospecific and polyspecific *in silico* experimental settings, the proposed probe design method selected a small number of oligo probes of length 7 or 8 nucleotides that perfectly classified all unseen testing sequences. These results well illustrate the utility of the proposed method in genotyping applications.

**Keywords:** oligo probes, microarrays, LAD, set covering, learning theory, optimization, viral pathogens.

## 1 Introduction

A microarray or a DNA chip is a small glass or silica surface bearing DNA probes. Probes are single stranded reverse transcribed mRNAs, each located at a specific spot of the chip for hybridization with its Watson-Crick complementary sequence in a target to form the double helix [1]. Microarrays currently use two forms of probes, namely, oligonucleotide (shortly, oligo) and cDNA, and have prevalently been used in the analysis of gene expression levels, which measures the amount of gene expression in a cell by observing the hybridization of mRNA to different probes, each targeting a specific gene. With the ability to identify a specific target in a biological sample, microarrays are also well-suited for detecting biological agents for genetic and chronic disease [2,3,4,5]. Furthermore, as viral pathogens can be detected at the molecular and genomic level much before the onset of physical symptoms in a patient, the microarray technology can be used for an early detection of patients infected with viral pathogens [6,7,8].

---

\* This paper is dedicated to the life and memory of Dr. Peter L. Hammer (December 23, 1936 - December 27, 2006), the inventor of LAD and an OR giant.

\*\* This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2005-003-D00445).

\*\*\* Corresponding author.

The success of microarrays depends on the quality of probes that are tethered on the chip. Having an optimized set of probes is beneficial for two reasons. One, the background hybridization is minimized, hence true gene expression levels can be more accurately determined [9]. The other, as the number of oligos needed per gene is minimized, the cost of each microarray is minimized or the number of genes on each chip is increased, yielding oligo fingerprinting a much faster and more cost-efficient technique [9,10]. Short probes consisting of 15-25 nucleotides (nt) are used in genotyping applications [1]. Having short optimal probes means a high genotyping accuracy in terms of both sensitivity and specificity [6,9] and can play a key role in genotyping applications.

From the perspective of numerical optimization, genomic data present an unprecedented challenge for supervised learning approaches for a number of reasons. First, genomic data are long sequences over the nucleic acid alphabet  $\Sigma = \{A, C, G, T\}$ . Second, for example, the complexity of viral flora, owing to constantly evolving viral serotypes, requires a supervised learning theory to be trained on a large collection of target and non-target samples. That is, a typical training set contains a large number of large-scale samples. Third, a supervised learning framework usually requires a systematic pairing or differencing between each target and non-target samples during the course of training a decision rule [10,11,12,13]. Adding to these, the nature of data classification is difficult [14].

Based on the general framework of logical analysis of data (LAD) from [15], we develop in this paper a probe design method for selecting short oligo probes of length  $l$  nt, where  $l \in [6, 10]$ . To list some advantages of selecting oligo probes by the proposed method, first, the method selects probes via sequential solution of a small number of compact set covering (SC) instances, which offers a great advantage from computational point of view. To be more specific, consider classification of two types of data and suppose that a training set is comprised of  $m^+$  target and  $m^-$  non-target sequences. The size of the SC training instances solved by the proposed method is minimum of  $m^+$  and  $m^-$  orders of magnitude smaller than optimization learning models used in [10,11,12]. Second, the method uses the sequence information only and selects probes via optimization based on principles of probability and statistics. That is, the probability of an  $l$ -mer (oligo of length  $l$ ) appearing in a single sequence by chance is  $(0.25)^l$ , hence the probability of an  $l$ -mer appearing in multiple samples of one type but in none or only a few of the sequences of the other type by chance alone is extremely small. Third, the proposed method does not rely on any extra tool, such as BLASTn [16], a local sequence alignment search tool that is commonly used for probe selection [6,8,17], or the existence of pre-selected representative probes [6]. This makes the method truly stand-alone and free of problems that may possibly be caused by limitations associated with external factors. Last, with an array of efficient (meta-)heuristic solution procedures for SC, the proposed method is readily implementable for an efficient selection of oligo probes.

As for the organization of this paper, we develop an effective method for selecting short oligo probes in Section 2 (for reasons of space, we omit proofs for the mathematical results in this section) and extensively test the proposed

probe design method in various *in silico* genotyping experiments in Section 3 with using viral genomic sequences from the Los Alamos National Laboratory and the National Center of Biotechnology Information websites.

## 2 Proposed Probe Selection Method

The task of classifying more than two types of data can be accomplished by sequential classifications of two types of + and – data (see [18,19,20] and Section 3 below). Without loss of generality, therefore, we present the material below in the context of binary classification.

The backbone of the proposed procedure is LAD. A typical implementation of LAD analyzes data on hand via four sequential stages of data binarization, support feature selection, pattern generation and classification rule formation. As a Boolean logic-based, LAD first converts all non-binary data into equivalent binary observations. A + (–) ‘pattern’ in LAD is defined as a conjunction of one or more binary attributes or their negations that distinguishes one or more + (–) type observations from all – (+) observations. The number of attributes used in a pattern is called the ‘degree’ of the pattern. As seen from the definition, patterns hold the structural information hidden in data. After patterns are generated, they are aggregated into a partially-defined Boolean discriminant function/rule to generalize the discovered knowledge to classify new observations.

Referring readers to [13,15,21] for more background in LAD, we design a LAD-based method below for efficiently analyzing large-scale genomic data.

### 2.1 Data Binarization

Let there be  $m^+$  and  $m^-$  sample observations of type + (target) and – (non-target), respectively. For  $\bullet \in \{+, -\}$ , let us use  $\bullet$  to denote the complementary element of  $\bullet$  with respect to the set  $\{+, -\}$ . Let  $S^\bullet$  denote the index set of  $m^\bullet$  sample sequences for  $\bullet \in \{+, -\}$ .

A DNA sequence is a sequence of nucleic acids A, C, G and T, and the training sequences need to be converted into Boolean sequences of 0 and 1 before LAD can be applied. Toward this end, we first choose an integer value for  $l$ , usually  $l \in [6, 10]$  (see Section 3), generate all  $4^l$  possible  $l$ -mers over the four nucleic acid letters and then number them consecutively from 1 to  $4^l$  by a mapping scheme. Next, each  $l$ -mer is selected in turn and every training sample is fingerprinted with the oligo for its presence or absence. That is, with oligo  $j$ , we scan each sequence  $p_i$ ,  $i \in S^+ \cup S^-$ , from the beginning of the sequence and shifting to the right by a base and stamp

$$p_{ij} = \begin{cases} 1, & \text{if oligo } j \text{ is present in sequence } i; \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

After this, the oligos that appear in all or none of the training sequences can be deleted from further consideration. We re-number the surviving  $l$ -mers consecutively from 1 to  $n$  and replace the original training sequences described in the nucleic acid alphabets by their Boolean representations. Let  $N = \{1, \dots, n\}$ .

## 2.2 Pattern Generation

The data are now described by  $n$  attributes  $a_j \in \{0, 1\}$ ,  $j \in N$ . For observation  $p_i$ ,  $i \in S^\bullet$ ,  $\bullet \in \{+, -\}$ , let  $p_{ij}$  denote the binary value the  $j$ -th attribute takes in this observation. Denote by  $l_j$  the literal of binary attribute  $a_j$ . Then,  $l_j = a_j$  ( $l_j = \bar{a}_j$ ) instructs to take (negate) the value of  $a_j$  in all sequences. A term  $t$  is a conjunction of literals. Given a term  $t$ , let  $N_t \subseteq N$  denote the index of literals included in the term. Then, we have  $t = \bigwedge_{j \in N_t} l_j$ . A  $\bullet$  pattern is a

term that satisfies  $t(p_i) := \prod_{\substack{l_j=a_j, \\ j \in N_t}} p_{ij} \prod_{\substack{l_j=\bar{a}_j, \\ j \in N_t}} \bar{p}_{ij} = 1$  for at least one  $p_i$ ,  $i \in S^\bullet$ , and  $t(p_k) = 0$  for all  $p_k$ ,  $k \in S^\bullet$ . Note here that  $N_t$  of a  $\bullet$  pattern identifies probes that collectively distinguish one or more  $\bullet$  sequences from the sequences of the other type.

Let us introduce  $n$  additional features  $a_{n+j}$ ,  $j \in N$ , and use  $a_{n+j}$  to negate  $a_j$ . Let  $N' = \{1, \dots, 2n\}$  and let us introduce a binary decision variable  $x_j$  for  $a_j$ ,  $j \in N'$ , to determine whether to include  $l_j$  in a pattern. [15] formulated a compact mixed integer and linear programming (MILP) model below with respect to a reference sample  $p_i$ ,  $i \in S^\bullet$ ,  $\bullet \in \{+, -\}$ :

$$\begin{array}{l|l}
 & \begin{array}{l} z_{2,i} = \min_{\mathbf{x}, \mathbf{y}, d} \sum_{l \in S^\bullet \setminus \{i\}} y_l \\ \text{s. t.} \quad \sum_{j \in J_i} x_j = d \\ \sum_{j \in J_i} p_{lj} x_j + y_l \geq d, \quad l \in S^\bullet \setminus \{i\} \\ \sum_{j \in J_i} p_{lj} x_j \leq d - 1, \quad l \in S^{\bar{\bullet}} \\ 1 \leq d \leq n \\ \mathbf{x} \in \{0, 1\}^n \\ \mathbf{0} \leq \mathbf{y} \leq \mathbf{n}, \end{array} \\
 \text{(MILP-2.i}^\bullet\text{)} & 
 \end{array}$$

where  $J_i := \{j \in N' : p_{ij} = 1\}$  for  $p_i$ ,  $i \in S^\bullet$ . Consider the following.

**Lemma 1.** *Let  $(\mathbf{x}, \mathbf{y}, d)$  denote a feasible solution of (MILP-2.i $^\bullet$ ). Let  $N_t = \{j \in J_i : x_j = 1\}$ . Then,  $\mathcal{P} := \bigwedge_{j \in J_i, x_j=1} a_j$  forms a  $\bullet$  pattern.*

We note here that genomic data are large-scale in nature. Furthermore, owing to constantly evolving viral serotypes, the complexity of viral flora is high, and this requires large numbers of target and non-target viral samples to be used for selecting optimal genotyping probes. Adding to these the difficulties associated with numerical solution of MILP, we see that (MILP-2.i $^\bullet$ ) above presents no practical way of selecting genotyping probes.

With the need to develop a more efficient pattern generation scheme, we select a reference sequence  $p_i$ ,  $i \in S^\bullet$ ,  $\bullet \in \{+, -\}$ , and set

$$a_j^{(i,k)} = \begin{cases} 1, & \text{if } p_{ij} \neq p_{kj}; \text{ and} \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

for  $k \in S^\bullet$  and  $j \in N$ . Next, we set

$$a_j^{(i,l)} = \begin{cases} 1, & \text{if } p_{ij} = p_{lj}; \text{ and} \\ 0, & \text{otherwise,} \end{cases}$$

for  $l \in S^\bullet$  and  $j \in N$ . Now, consider the set covering model

$$(SC_i^\bullet) \quad \left| \begin{array}{ll} \min_{\mathbf{x}, \mathbf{y}} & \sum_{j \in N} c_j x_j + \sum_{l \in S^\bullet \setminus \{i\}} y_l \\ \text{s.t.} & \sum_{j \in N} a_j^{(i,l)} x_j + y_l \geq 1, \quad l \in S^\bullet \setminus \{i\} \\ & \sum_{j \in N} a_j^{(i,k)} x_j \geq 1, \quad k \in S^\bullet \\ & x_j \in \{0, 1\}, \quad j \in N \\ & y_l \in \{0, 1\}, \quad l \in S^\bullet \setminus \{i\}, \end{array} \right.$$

where  $c_j$  ( $j \in N$ ) are positive real numbers.

**Theorem 1.** Let  $(\mathbf{x}, \mathbf{y})$  denote a feasible solution of  $(SC_i^\bullet)$ . Then,

$$\mathcal{P} := \bigwedge_{\substack{x_j=1, \\ p_{i\bullet}^\bullet=1}} a_l \bigwedge_{\substack{x_l=1, \\ p_{i\bullet}^\bullet=0}} \bar{a}_l \quad (2)$$

forms a  $\bullet$  LAD pattern.

**Lemma 2.** With a feasible solution  $(\mathbf{x}, \mathbf{y})$  of  $(SC_i^\bullet)$ , let  $N_t = \{j \in N : x_j = 1\}$ . Then,  $y_l = 0$  for  $l \in S^\bullet \setminus \{i\}$  if and only if  $p_{lk} = p_{ik}$  for all  $k \in N_t$ .

Although smaller than the MILP counterpart by only one constraint and one integer variable,  $(SC_i^\bullet)$  has a much simpler structure and is defined only in terms of 0-1 variables. In addition, it can exploit any of SC heuristic procedures developed so far (see, for example, [22] and references therein) for its efficient solution, hence is much preferred.

Note that  $(SC_i^\bullet)$  is defined by  $m^+ + m^- - 1$  cover inequalities and  $n + m^\bullet - 1$  binary variables. Also, recall that  $n$  is large for genomic sequences and the analysis of viral sequences requires large numbers of target and non-target sequences, that is,  $m^+$  and  $m^-$  are also large numbers. To develop a more compact SC-based probe selection model, we select a reference sequence  $p_i$ ,  $i \in S^\bullet$ ,  $\bullet \in \{+, -\}$ , and

set the values of  $a_j^{(i,k)}$  for  $k \in S^\bullet$  and  $j \in N$  via (1). Consider the following SC model:

$$(SC\text{-}pg_i^\bullet) \quad \left| \begin{array}{l} \min_{\mathbf{x}} \sum_{j \in N} c_j x_j \\ \text{s.t.} \sum_{j \in N} a_j^{(i,k)} x_j \geq 1, \quad k = 1, \dots, m^\bullet \\ x_j \in \{0, 1\}, \quad j \in N, \end{array} \right.$$

where  $c_j$ 's are positive reals.

**Theorem 2.** *Let  $\mathbf{x}$  denote a feasible solution of  $(SC\text{-}pg_i^\bullet)$ . Then,  $\mathcal{P}$  generated on  $\mathbf{x}$  via (2) forms a  $\bullet$  LAD pattern.*

**Lemma 3.** *With a feasible solution  $\mathbf{x}$  of  $(SC_i^\bullet)$ , generate a  $\bullet$  pattern  $\mathcal{P}$  via (2). Then,  $\mathcal{P}$  distinguishes every  $\bullet$  sequence  $p_l$ ,  $l \in S^\bullet$ , with  $p_{lk} = p_{ik}$  for all  $k \in N_t$  from the  $\bar{\bullet}$  observations, where  $N_t = \{j \in N : x_j = 1\}$ .*

Below, we use  $(SC\text{-}pg_i^\bullet)$  to design one simple oligo probe selection procedure. Let  $P^\bullet$  denote the set of  $\bullet$  patterns generated so far.

**procedure SC-pg**

**begin**

**for**  $\bullet \in \{+, -\}$  **do**

    set  $P^\bullet = \emptyset$  and  $S \leftarrow S^\bullet$ .

**while**  $S \neq \emptyset$  **do**

      - randomly choose  $p_i$ ,  $i \in S$ , and formulate  $(SC\text{-}pg_i^\bullet)$ .

      - solve  $(SC\text{-}pg_i^\bullet)$ .

      - generate a  $\bullet$  pattern  $\mathcal{P}$  via (2).

      - set  $P^\bullet = P^\bullet \cup \{\mathcal{P}\}$  and  $S = S \setminus \{i\} \setminus \{j \in S, j \neq i : p_{jk} = p_{ik}, \forall k \in N_t\}$ .

**end while**

**end for**

**end**

**Theorem 3.** *procedure SC-pg terminates finitely.*

### 3 Experiments and Discussions

In this section, we extensively test the proposed probe design for the classification of viral disease-agents in *in silico* setting with using genomic sequences obtained from the Los Alamos National Laboratory (LANL) and the National Center for Biotechnology Information (NCBI). Table 1 summarizes the number and the length (the minimum, average $\pm$ 1 standard deviation and maximum lengths) of each type of the genomic data that were used in our experiments.

In analyzing data in an experiment, we first decided on a length of oligos to use by calculating the smallest integer value  $l$  such that  $4^l$  became larger than or equal to the average of the lengths of target and non-target sequences of the experiment. Then,  $4^l$  candidate oligos were generated to fingerprint and binarize

the data. Note here that if a constraint in  $(SC\text{-}pg_i^\bullet)$  has all zero coefficients, then the SC instance has no feasible solution, and this case arises when the reference sequence  $p_i$ ,  $i \in S^\bullet$ , and the sequence  $p_j$ ,  $j \in S^\bullet$  have identical 0-1 fingerprints, which is a contradiction. Supervised learning methodologies, including LAD, presume for the existence of a classification function that each unique sequence in the training set belongs to exactly one of the two classes. When data under analysis are indeed contradiction-free, then contradiction-free 0-1 clones of the data can always be obtained by using oligos of longer length for data fingerprinting and binarization. Therefore, when we generated the identical fingerprint for data of different types, we incremented the value of  $l$  by 1 and repeated the data binarization stage until the binary representations of the data became contradiction free. Next, **procedure**  $SC\text{-}pg$  was applied to generate patterns, hence probes. In applying **procedure**  $SC\text{-}pg$  in these *in silico* experiments, we selected a minimal set of oligo probes by setting  $c_j = 1$  for all  $j \in N$ . For solving the unicast  $(SC\text{-}pg_i^\bullet)$ 's generated, we used the textbook greedy heuristic [23] for ease of implementation.

Denote by  $P_1^+, \dots, P_{n_+}^+$  and  $P_1^-, \dots, P_{n_-}^-$  the positive and negative patterns, respectively, generated via **procedure**  $SC\text{-}pg$ . In classifying unseen  $+$  (target) and  $-$  (non-target) sequences, we use three decision rules. Specifically, for the polyspecific genotyping experiments (in Section 3.1 and Experiments 2 and 3 in Section 3.2), we form the standard LAD classification rule [13]

$$\Delta := \sum_{i=1}^{n_+} \frac{\omega_i^+}{|S^+|} P_i^+ - \sum_{i=1}^{n_-} \frac{\omega_i^-}{|S^-|} P_i^-, \quad (3)$$

where  $\omega_i^\bullet$  denotes the number of  $\bullet$  training sequences covered by  $P_i^\bullet$ . We assign class  $+$  ( $-$ ) to new sequence  $p$  if  $\Delta(p) > 0$  ( $\Delta(p) < 0$ ). We fail to classify sequence  $p$  if  $\Delta(p) = 0$ .

For monospecific genotyping in Experiment 1 in Section 3.2, we form a decision rule by

$$\Delta^+ := \sum_{i=1}^{n_+} P_i^+ \text{ and } \Delta^- := \sum_{i=1}^{n_-} P_i^- \quad (4)$$

and assign  $p$  to class  $\bullet$  if  $\Delta^\bullet(p) > 0$  while  $\Delta^{\bar{\bullet}}(p) = 0$ . When  $\Delta^\bullet(p) > 0$  and  $\Delta^{\bar{\bullet}}(p) > 0$  or when  $\Delta^\bullet(p) = 0$  and  $\Delta^{\bar{\bullet}}(p) = 0$ , we fail in classifying the sequence.

For monospecific classification of more than two viral (sub-)types  $k = 1, \dots, m$  in Experiment 4 in Section 3.2, we use the decision rule

$$\Delta^k := \sum_{i=1}^{n_k} P_i^k, \quad (5)$$

where  $P_1^k, \dots, P_{n_k}^k$  are the probe(s) selected to for virus (sub-)type  $k$ , and assign  $p$  to class  $k$  if  $\Delta^k(p) > 0$  while  $\Delta^i(p) = 0$  for all  $i = 1, \dots, m, i \neq k$ . When  $\Delta(p) > 0$  for more than two virus types or  $\Delta^k(p) = 0$  for all  $k$ , then we fail to assign a class to sequence  $p$ .

In each of the experiments in this section, we tested the proposed oligo probe selection method in 20 independent hold-out experiments, each with randomly selected 90% of the target and of the non-target data forming a training set of sequences and the remaining 10 % of the target and of the non-target sequences forming the testing data. More specifically, after a training set of data was formed, we binarized the training data and selected optimal oligo probes on them via **procedure SC-pg**. Next, a classification rule was formed by one of (3), (4) and (5) above and then used for classifying the corresponding testing sequences. These steps were repeated 20 times to obtain the average testing performance and other relevant information of the experiment.

The computational platform used for these experiments was an Intel 2.66GHz Pentium Linux PC with 512Mb of memory.

**Table 1.** Viral sequences used in experiments

viral sequence	number	length		
		min.	avg. $\pm$ 1 std. dev.	max.
human papillomavirus (HPV):				
- high risk HPV	18	449	7365 $\pm$ 1730	7989
- low risk HPV	54	455	7198 $\pm$ 1683	8027
SARS coronavirus	105	29350	29692 $\pm$ 91	29765
coronavirus	39	9203	29013 $\pm$ 3569	31526
other virus:				
- human respiratory syncytial virus	10	13933	15091 $\pm$ 386	15226
- human adenovirus	32	34125	35215 $\pm$ 618	36015
- human parainfluenza virus	4	15646	15652 $\pm$ 3	15654
- human rhinovirus (A, B)	8	7102	7157 $\pm$ 36	7212
- influenza virus (A, B, C)	53	838	1701 $\pm$ 527	2368
influenza virus hemagglutinin (H) subtype:				
- H1	137	1698	1749 $\pm$ 24	1778
- H3	660	1695	1735 $\pm$ 21	1768
- H5	148	1677	1721 $\pm$ 25	1779
- H7	77	1659	1690 $\pm$ 27	1792
- H9	93	1683	1704 $\pm$ 26	1742
- H else (2, 4, 6, 8, 11, 12, 13, 16)	65	1689	1742 $\pm$ 29	1773
influenza virus neuraminidase (N) subtype:				
- N1	218	1344	1410 $\pm$ 39	1463
- N2	1050	1341	1434 $\pm$ 28	1467
- N3	44	1326	1411 $\pm$ 29	1460
- N else (4, 5, 6, 7, 8, 9)	64	1341	1434 $\pm$ 25	1467

### 3.1 Classification of High and Low Risk HPV: A Comparative Experiment

The infection with HPV is the main cause of cervical cancer, the second most common cancer in women worldwide [24,25]. There are more than 80 identified



types of HPV and the genital HPV types are subdivided into high and low risk types: low risk HPV types are responsible for most common sexually transmitted viral infections while high risk HPV types are a crucial etiological factor for the development of cervical cancer [26].

We applied the proposed probed design method on the 72 HPV sequences downloaded from LANL with their classification found in Table 3 of [27]. The selected probes were used to form a decision rule by (3) and tested for their classification capability.

Results from this polyspecific probe selection experiment are provided in Table 2. In this table and also in the table found in the following subsection, the target (+) and the non-target (−) virus types of the experiments are first specified. Then, the tables provide two bits of information on the candidate oligos, namely, the length  $l$  and the average and the standard deviation of the number of features generated and used in the 20 runs of each experiment for data binarization and for pattern generation. Provided next in the tables is the information on the number of probes selected in the format ‘the average  $\pm$  1 standard deviation’ and information on the LAD patterns generated. Finally, the testing performance of the probes selected is provided in the last column of the tables, summarized in format ‘the average  $\pm$  1 standard deviation’ of the percentage of the correct classifications of the unseen sequences.

**Table 2.** Polyspecific classification of high and low risk HPV by the proposed method

Experiment	$l$ -mers used		probes selected		testing accuracy <sup>*†</sup>
	$l$	number <sup>*</sup>	number <sup>*</sup>	patterns	
high risk HPV (+) vs. HPV low (−)	8	58359.9 $\pm$ 130.4	18.7 $\pm$ 1.7 22.8 $\pm$ 1.6	degree 1 & 2 patterns degree 1 & 2 patterns	90.6 $\pm$ 9.8

<sup>\*</sup>: in format average  $\pm$  standard deviation

<sup>†</sup>: percentage of correct classifications of testing/unseen data

Briefly summarizing, the proposed probe design method selected probes on the HPV data in a few CPU seconds that tested 90.6% accurate in classifying the unseen HPV samples. For comparison, the same HPV dataset was used in [2] and [27] for the classification of HPV by high and low risk types. In brief, the probe design methods of [2] and [27] required several CPU hours of computation and selected probes that obtained 85.6% and 81.1% correct classification rates, respectively.

Before moving on, we note that the sequences belonging to the target and the non-target groups in this experiment all have different HPV subtypes (see Table 3 in [27]). The combination of all target and non-target sequences being different from one another and the presence of noise in the data (the classification errors) gave rise to selecting a relatively large number of polyspecific probes in this experiment.

### 3.2 Genotyping Experiments with Viral Pathogens

The proposed probe design method was tested on genomic viral sequences from NCBI for selecting monospecific and polyspecific probes for screening for SARS and AI in a number of different binary and multicategory experimental setting and performed superbly on all counts. We describe individual experiments below and summarize results from these experiments in Table 3.

**Table 3.** Genotyping viral pathogens by the proposed method

Experiment	viruses	$l$	$l$ -mers used	probes selected		testing
	distinguished			number*	patterns generated	accuracy*†
1	SARS virus (+)	8	$57745.3 \pm 306.1$	$1 \pm 0$	degree 1	$100 \pm 0$
	coronavirus (-)			$1 \pm 0$	degree 1	
2	SARS virus (+)	8	$64141.5 \pm 36.5$	$1 \pm 0$	degree 1	$100 \pm 0$
	influenza virus (-)			$10.1 \pm 0.8$	degree 1 only	
3	H5 & H9 (+)	8	$39056 \pm 398.3$	$6.7 \pm 0.5$	degree 1 only	$100 \pm 0$
	other H strains (-)			$21.6 \pm 1.3$	degree 1 only	
4	N1	7	$13151 \pm 39.3$	$3 \pm 0$	degree 1	$100 \pm 0$
	N2			$3.7 \pm 0.5$	degree 1 only	
	N3			$1 \pm 0$	degree 1	

\*: in format average  $\pm$  standard deviation

†: percentage of correct classifications of testing/unseen data

#### Experiment 1. SARS virus vs. coronavirus

SARS virus is phylogenetically most closely related to group 2 coronavirus [28]. 105 SARS sequences and 39 coronavirus samples were used to select 1 monospecific probe for screening for SARS. Used in a classification rule (4), the SARS probe and one probe selected for coronavirus together perfectly classified all testing sequences.

#### Experiment 2. SARS virus vs. influenza virus

This experiment simulates a SARS pandemic where suspected patients with SARS-like symptoms are screened for the disease. We used the 105 SARS virus sequences and 108 samples of other influenza virus types (the ‘other virus’ in Table 1) in this experiment and selected polyspecific probes. Used in a classification rule (3), these probes collectively gave the perfect classification of all testing sequences.

#### Experiment 3. Classification of lethal AI virus H5 & H9 and other influenza virus H subtypes

AI virus H5 and H9 subtypes cause a most fatal form of the disease [29], and they were separated from the other H subtypes of influenza virus in this experiment. 241 H5 and H9 target sequences and 1010 other H subtype sequences were used to select polyspecific probes for detecting AI virus H5 and H9 subtypes from the rest. In a classification rule (3), the selected probes collectively classified all testing sequences correctly.

#### **Experiment 4.** Monospecific Classification of N1, N2 and N3 influenza virus

The statement “monospecific neuraminidase (NA) subtype probes were insufficiently divers to allow confident NA subtype assignment” from [6] motivated us to design this experiment on multicategory and monospecific classification of influenza virus by N subtypes. We used the three influenza virus N subtypes with 30 or more samples in Table 1 and selected monospecific probes for their classification. Tested in a classification rule (5), the selected probes performed perfectly in classifying all testing sequences. Note that only a small number of monospecific probes were selected and proved ‘needed’ in this experiment.

## **References**

1. Stears, R., Martinsky, T., Schena, M.: Trends in microarray analysis. *Nature Medicine* **9**(1) (2003) 140–145
2. Eom, J.H., Park, S.B., Zhang, B.T.: Genetic mining of dna sequence structures for effective classification of the risk types of human papillomavirus (hpv). In Pal, N., Kasabov, N., Mudi, R., Pal, S., Parui, S., eds.: *Lecture Notes in Computer Science*. Volume 3316. Springer-Verlag, Berlin Heidelberg (2004) 1334–1343
3. Heller, R., Schena, M., Chai, A., Shalon, D., Bedilion, T., Gilmore, J., Woolley, D., Davis, R.: Discovery and analysis of inflammatory disease-related genes using cdna microarrays. *Proceedings of the National Academy of Sciences* **94** (1997) 2150–2155
4. Liu, C.H., Ma, W.L., Shi, R., Ou, Y.Q., Zhang, B., Zheng, W.L.: Possibility of using dna chip technology for diagnosis of human papillomavirus. *Journal of Biochemistry and Molecular Biology* **36**(4) (2003) 349–353
5. Lee, Y., Lee, C.K.: Classification of multiple cancer types by multicategory support vector machines using gene expression data. *Bioinformatics* **19**(9) (2003) 1132–1139
6. Sengupta, S., Onodera, K., Lai, A., Melcher, U.: Molecular detection and identification of influenza viruses by oligonucleotide microarray hybridization. *Journal of Clinical Microbiology* **41**(10) (2003) 4542–4550
7. Vernet, G.: Dna-chip technology and infectious diseases. *Virus Research* **82** (2002) 65–71
8. Wang, D., Coscoy, L., Zylberberg, M., Avila, P., Boushey, H., Ganem, D., DeRisi, J.: Microarray-based detection and genotyping of viral pathogens. *PNAS* **99**(24) (2002) 15687–15692
9. Li, F., Stormo, G.: Selection of optimal dna oligos for gene expression arrays. *Bioinformatics* **17**(11) (2001) 1067–1076
10. Borneman, J., Chrobak, M., Vedova, G., Figueroa, A., Jiang, T.: Probe selection algorithms with applications in the analysis of microbial communities. *Bioinformatics* **17**(Suppl. 1) (2001) S39–S48
11. Rahmann, S.: Fast large scale oligonucleotide selection using the longest common factor approach. *Journal of Bioinformatics and Computational Biology* **1**(2) (2003) 343–361
12. Klau, G., Rahmann, S., Schliep, A., Vingron, M., Reinert, K.: Optimal robust non-unique probe selection using integer linear programming. *Bioinformatics* **20** (Suppl. 1) (2004) i186–i193
13. Boros, E., Hammer, P., Ibaraki, T., Kogan, A., Mayoraz, E., Muchnik, I.: An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering* **12** (2000) 292–306

14. Megiddo, N.: On the complexity of polyhedral separability. *Discrete and Computational Geometry* **3** (1988) 325–337
15. Ryoo, H., Jang, I.Y.: Milp approach to pattern generation in logical analysis of data. *Machine Learning* (2005) submitted.
16. Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic local alignment search tool. *Journal of Molecular Biology* **215** (1990) 403–410
17. Wang, X., Seed, B.: Selection of oligonucleotide probes for protein coding sequences. *Bioinformatics* **19**(7) (2003) 796–802
18. Cortes, C., Vapnik, V.: Support vector networks. *Machine Learning* **20** (1995) 273–297
19. Ullman, J.: *Pattern Recognition Techniques*. Crane, London (1973)
20. Vapnik, V.: *Statistical Learning Theory*. Wiley-Interscience (1998)
21. Hammer, P.: Partially defined boolean functions and cause-effect relationships. *Proceedings of the International Conference on Multi-Attribute Decision Making Via OR-Based Expert Systems* (1986)
22. Caprara, A., Fischetti, M., Toth, P.: A heuristic method for the set covering problem. *Operations Research* **47**(5) (1999) 730–743
23. Nemhauser, G.L., Wolsey, L.A.: *Integer and Combinatorial Optimization*. Wiley-Interscience Series I Discrete Mathematics and Optimization. Wiley, New York (1988)
24. Muñoz, N., Bosch, F., de Sanjosé, S., Herrero, R., Castellsagué, X., Shah, K., Snijders, P., C.J.L.M. Meijer, for the International Agency for Research on Cancer Multicenter Cervical Cancer Study Group: Epidemiologic classification of human papillomavirus types associated with cervical cancer. *The New England Journal of Medicine* **348**(6) (2003) 518–527
25. Bosch, F., Lorincz, A., Muñoz, N., Meijer, C., Shah, K.: The causal relation between human papillomavirus and cervical cancer. *Journal of Clinical Pathology* **55** (2002) 244–265
26. McFadden, S., Schumann, L.: The role of human papillomavirus in screening for cervical cancer. *Journal of the American Academy of Nurse Practitioners* **13** (2001) 116–125
27. Park, S.B., Hwang, S.H., Zhang, B.T.: Classification of the risk types of human papillomavirus by decision trees. In: *Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning*. (2003) 540–544
28. Snijder, E., Bredenbeek, P., Dobbe, J., Thiel, V., Ziebuhr, J., Poon, L., Guan, Y., Rozanov, M., Spaan, W., Gorbalenya, A.: Unique and conserved features of genome and proteome of sars-coronavirus, an early split-off from the coronavirus group 2 lineage. *Journal of Molecular Biology* **331** (2003) 991–1004
29. Koopmans, M., Wilbrink, B., Conyn, M., Natrop, G., van der Nat, H., Vennema, H., Meijer, A., van Steenbergen, J., Fouchier, R., Osterhaus, A., Bosman, A.: Transmission of h7n7 avian influenza a virus to human beings during a large outbreak in commercial poultry farms in the netherlands. *Lancet* **363** (2004) 587–593 [www.thelancet.com](http://www.thelancet.com).

# Learning the Semantic Meaning of a Concept from the Web

Yang Yu and Yun Peng

Department of Computer Science and Electrical Engineering,  
University of Maryland Baltimore County, Baltimore, MD 21250, USA  
{yangyu1, ypeng}@umbc.edu

**Abstract.** Many researchers have used text classification method in solving the ontology mapping problem. Their mapping results heavily depend on the availability of quality exemplars used as training data. However, manual preparation of exemplars is costly. In this work, we propose to automatically extract text from web pages returned by a search engine. Search queries are formed according to the semantic information given in the ontology. We have implemented a prototype system that automates the entire process (from search query formation to conditional probability calculation) and conducted a series of experiments. We assessed the effectiveness of our approach by comparing the obtained conditional probabilities with human expectations. Our main contribution is that we explored the possibilities of utilizing web information for text classification based ontology mapping and made several valuable discoveries on its usefulness for future research.

**Keywords:** Semantic web, ontology mapping, text classification, search engine.

## 1 Introduction

The semantic web is an "extension of the current web" [1], where information is marked up by ontology languages such as OWL and RDF so that it can be understood and processed by programs. However, it is not realistic to assume everyone shares a single ontology. Instead, different organizations may have different ontologies for the same domain, reflecting their designers' own perceptions and conceptualizations of the domain. For example, a course on neural networks may be called "Introduction to Neural Networks" in one university's course ontology but "Introduction to Connectionist Models" in another's. Understanding these two courses actually teaching similar materials will not be a problem to a computer science professor because in the professor's knowledge base, the two course titles have the same or very similar meaning or semantics. However, when programs based on one ontology try to exchange information with programs based on another, problems will happen. This so-called interoperability problem has been known for a long time in software integration, and becomes more acute in the semantic web [2].

One of the approaches to address this interoperability problem is to map concepts defined in one ontology to semantically identical or similar concepts in another. Text classification is a very powerful technique some have suggested for this purpose

[8, 9]. However, its success is highly dependent on the availability of text documents that are exemplars of individual concepts in the ontologies. Manually preparing a good number of exemplars for hundreds of concepts is time-consuming and very costly. This greatly reduces the attractiveness of text classification based ontology mapping. To address this difficulty, we propose to automatically retrieve exemplars from the web, the largest information source available. A prototype system has been built based on this idea, which allows us to experiment with different parameters and methods in each step of this approach. A series of experiments have shown encouraging results.

The rest of the paper is organized as follows. Section 2 provides background and motives of this work; Section 3 presents the technical details of this approach and the prototype system; Section 4 describes the experiments and results; Section 5 discusses related works; and Section 6 concludes with suggestions to future research.

## 2 Background and Motivation

In computer science, an ontology is a set of concepts each of which can have individual members, its own properties, and its relations with other concepts in the set. For example, Fig.1 shows a simple ontology defined in OWL based on [3].

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Class rdf:ID= "CommercialJet"></rdf:Class>
<rdf:Class rdf:ID= "BoeingJet">
  <rdf:subClassOf rdf:resource ="CommercialJet"/></rdf:Class>
<rdf:Class rdf:ID= "AirbusJet">
  <rdf:subClassOf rdf:resource ="CommercialJet"/></rdf:Class>
<rdf:Class rdf:ID= "Boeing-747">
  <rdf:subClassOf rdf:resource ="BoeingJet"/></rdf:Class>
<rdf:Class rdf:ID= "A-380">
  <rdf:subClassOf rdf:resource ="AirbusJet"/></rdf:Class>
</rdf:RDF>
```

**Fig. 1.** Ontology for CommercialJet in OWL

From this ontology, we know, by the *SubClassOf* property, that Boeing-747 is a kind of Boeing Jet, which itself is a kind of commercial jet. If we define a “made-in” property, we can specify “Boeing-Jets are *made-in* WA”. If the ontology also has information that WA is the same as Washington State, and it is *a-part-of* USA, then these data can be easily used by a program to answer questions like “Find all types of commercial jets that are made in the USA”.

By defining relations between concepts, we effectively build a web of concepts, potentially a huge integrated database, where information can be shared among applications easily and more complicated reasoning can be supported. The arrival of this semantic web requires ontologies to be developed and shared by many organizations and individuals. However, it is hard to make ontology development a coordinated centralized activity and it is also a fact that people can use different terms for one concept or similar concepts, so different ontologies can be created for the

same domain. For the semantic web to work, it is imperative to relate or map concepts between such ontologies.

Different approaches to ontology mapping have been developed. Manual mappings between large ontologies have been tried in recent years [4, 5]. The mapping is accurate and it can be saved for future use. The problem is that the size of ontologies can be very large and ontologies can keep growing, which requests a huge amount of continuous human efforts in establishing and maintaining the mapping. Consequently more researchers are looking for ways to map ontologies (semi)automatically.

String matching of concept names in two ontologies [6] is an effective alternative. Large amount of information can be processed very quickly and with a high degree of accuracy. For example, “meetingPlace” and “PlaceOfMeeting” can be matched. But matching “Tank” and “Armored Motor Vehicle” would usually involve complicated lexical analysis, and a complete dictionary such as WordNet has to be consulted.

Many researchers choose more powerful methods of machine learning, especially text classification techniques [7, 8, 9]. Usually, text exemplars for each concept or class in a given ontology ( $\text{Onto}_A$ ) are manually collected. Then a text classifier is trained using these data. To map a concept  $C$  defined in another ontology ( $\text{Onto}_B$ ) to some concept in  $\text{Onto}_A$ , exemplars for  $C$  need to be collected and classified into the classifier of  $\text{Onto}_A$ . Based on the initial classification results, algorithms such as [7] and [8] can be used to carry out the further steps of ontology mapping. Text classification based ontology mapping is much less time-consuming than manual mapping, and more powerful than string matching, because semantic meanings of apparently different strings can be analyzed by processing information contained in the provided exemplars. Here, the existence of exemplars for each concept and their relevancy to the concept they represent are the key factors to the effectiveness of this approach. However, finding sufficient, high-quality exemplars manually is costly, and is thus the limiting factor of this approach.

The WWW is the richest information resource available anywhere in the world. Collecting text exemplars from the WWW is a promising approach. To assess its effectiveness, we designed a tool to retrieve documents from the web through a search engine and tried a number of different ways to process the documents downloaded before using them for text classification. We tested our tool by actually performing some preliminary ontology mapping experiments with small-scale ontologies.

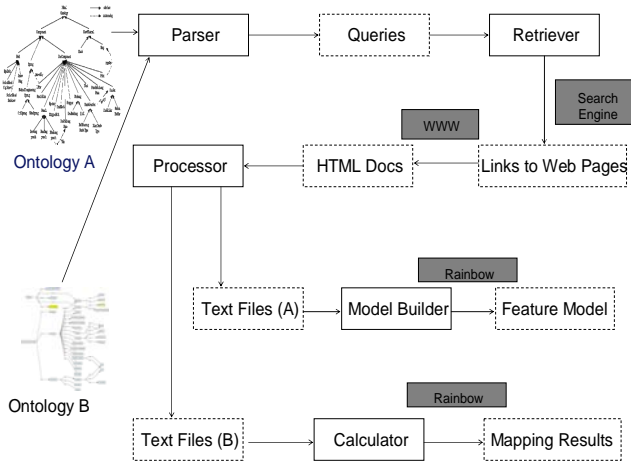
### 3 System Design

Here we use  $\text{Onto}_A$  to refer to the ontology in which we seek a mapping for a foreign concept and use  $\text{Onto}_B$  to refer to the ontology that provides the foreign concept. The system has the following main components:

1. A *parser* to parse ontology files in OWL format and to form search queries.
2. A *retriever* to drive a web search engine with the queries generated by the parser and to retrieve a specified number of web pages based on the search results.
3. A *processor* to process the raw HTML documents obtained from the retriever to construct text files as exemplars for concepts in the ontologies.
4. A *model builder* to build a probabilistic model for  $\text{Onto}_A$  from its exemplars using a text classification software. This model becomes  $\text{Onto}_A$ 's classifier.

5. A *calculator* to feed the text exemplar for concepts in  $\text{Onto}_B$  to  $\text{Onto}_A$ 's classifier, collects classification outputs and calculates conditional probabilities as initial mapping results.

We chose Google as our search engine and Rainbow [10] as our text classification software. The structure of the system is shown in Fig.2.



**Fig. 2.** System components overview

### 3.1 The Parser

We parse the ontology file to generate search queries for Google. To obtain better results, the query should contain more semantic information than just a class name. Because a word may have multiple senses or meanings, a query consisting of only the words of a concept's name may return web pages based on a more popular meaning of the word, which sometimes is not the particular meaning intended for the concept in the ontology. For example, in an ontology for food with a root class called "food", we may have a concept called apple, which is a subclass of "fruit". If we only use "apple" as the keyword, documents showing how to make an apple pie and documents showing how to use an iPod may both be returned. Apparently, the documents using apple as a computer manufacturing are irrelevant to a subclass of fruit. To avoid this, when forming a query, we use *all the terms on the path from the root class to the class in question* together as a query to send to the search engine. In the apple example, the query would be "food+fruit+apple" instead of "apple" itself alone. By doing so, the number of irrelevant documents returned is greatly reduced. This kind of "word sense disambiguation" by adding additional semantically relevant terms into the search queries can be further extended to include concept's properties. However, it needs to be noted that queries that include too many terms of high specificity (e.g., those in zoology or botany) may lead to very few search results.



### 3.2 The Retriever

The retriever takes a file containing queries generated by the parser, initiates a connection with a search engine, and sends a query in. It then goes through the search result pages for the query one by one and extracts URLs from each result page. After it collects a pre-specified number of URLs from the search results, it tries to download web pages at these URLs. Currently, only URLs starting with `http://` and ending with `.html`, `.htm` or `/` are extracted because other file types, for example, `doc` or `pdf` will be difficult for the processor to process. All the HTML files obtained through a query for a particular class are saved in one directory and will be used by the processor to generate exemplars for that class.

For most of the experiments, we retrieved documents using Google as the search engine, because it is the easiest one to be integrated into our system and it is generally considered the best. Although Google provides a programming API to obtain search results, we decided to develop our own retriever program. This gives us the flexibility to experiment with search engines other than Google (for example, Clusty.com) in some of our experiments.

### 3.3 The Processor

Documents collected by the retriever are HTML files. These raw data have to be processed before being used as exemplars. The processor will remove all HTML tags, image files, script programs, etc. Also removed are hyperlinks, which may contain some useful semantic information, but more often are just links to other irrelevant pages or websites. Since the retriever can easily retrieve a huge amount of relevant documents from the web, we can afford to be more selective in the process.

After the above steps, we have a large number of pure text files for each concept. The processor can perform some optional tasks: keeping only the sentences where a word in the query appears in each text file. Since not every part of a text file is necessarily relevant to the concept in question, this step may help remove irrelevant information and keep only the most closely relevant text. Text files processed with and without this option are both used in our experiments and the results are compared. The processor can also choose to keep paragraphs, rather than sentences in which query words appear.

### 3.4 The Model Builder

The system takes Naïve Bayes text classification approach to build a probabilistic model for concepts in  $\text{Onto}_A$ . In a text classification problem, we need to decide among a set of mutually exclusive categories  $C_1, C_2, C_3 \dots C_n$ , to which category a new document  $d$  should belong. This can be determined by which category has the greatest posterior probability, given  $d$ , i.e.,  $\max_i(P(C_i | d) | i = 1, \dots, n)$ , or equivalently,  $\max_i(P(d | C_i) * P(C_i))$  since only thing that matters here is the ranks among these categories.

Naïve Bayes approximates  $P(d | C_i)$  as follows. Let  $d$  contain  $m$  distinct words  $d = \{w_1, \dots, w_m\}$ , where  $w_i$  is the frequency of the  $i^{\text{th}}$  word in  $d$ . Then assuming that whether a word appears in a category is independent of other words in that category, we have the Naïve Bayes rule

$$P(d | C_i) = P(w_1, \dots, w_m | C_i) = \prod_{j=1}^m P(w_j | C_i)$$

Note that the independence assumption does not hold in general. Despite of this, good performance is still achieved. Details of this method can be found in [11].

A Naïve Bayes classifier requires the predefined categories  $C_1, C_2, C_3 \dots C_n$  to be mutually exclusive and exhaustive, so that the probability results can be correct and sum to 1. In our system, classification categories are closely related to ontology concept classes. Our model builder allows one to select concept classes in different ways when forming these classification categories.

For simplicity, this work only considers OWL ontology files that can be viewed as a concept tree based on the `subClassOf` property. The leaf classes in such a tree are assumed to be mutually exclusive and exhaustive regarding to the root class. By leaf classes, we mean those classes that do not have a subclass. The default behavior of the model builder is to use all the leaf classes in an ontology as the classification categories. Then Rainbow is called to build a probabilistic model for these categories.

Besides the default behavior, the model builder has an option to build a model for each class in  $\text{Onto}_A$  except the root. Two categories  $A^+$  and  $A^-$  are created by the model builder for class  $A$  in  $\text{Onto}_A$ .  $A^+$  is associated with exemplars for that class, and  $A^-$  is associated with exemplars for the complement of that class, which are taken from classes that are not  $A$ , not  $A$ 's ancestors nor  $A$ 's successors in the ontology tree. The model builder then builds a model using the exemplars for the two categories. This option is not applicable to the root class, because the root's complement is empty. For example, the exemplars for "not CAT" in the ontology tree shown in Fig.3 would include those found for all classes except "CAT" and its ancestors, "ANIMAL" and "LIVING\_THINGS".

### 3.5 The Calculator

Rainbow and other naïve Bayes text classifiers tend to produce extreme values (0 or 1) because of the independence assumption. This is certainly good enough if one only wants to get a right classification. However, our purpose is to use the classifier to obtain  $P(A | B)$  of concept  $A$  in  $\text{Onto}_A$ , given concept  $B$  in  $\text{Onto}_B$ , and hope to use this value as a basis to measure the semantic similarity between  $A$  and  $B$ . The calculator solves this problem by providing estimates of true conditional probabilities. It works as follows: (1) feeds all exemplars of concept  $B$  of  $\text{Onto}_B$  one by one to Rainbow, which performs classification using the model of  $\text{Onto}_A$ , (2) keeps records of the classification results for each exemplar, (3) calculate average results grouped by categories in the model as the conditional probability, and (4) write a summary report. It can also perform some additional calculations like estimating conditional probabilities for mapping involving non-leaf classes.

To see how classification results for a concept is averaged, suppose that APC is a class of  $\text{Onto}_B$ , a weapons ontology. For simplicity, suppose  $\text{Onto}_A$ , another weapons ontology, has three leaf classes: TANK-VEHICLE, AIR-DEFENSE-GUN, and SAUDI-NAVAL-MISSILE-CRAFT. We build a model using these three classes as classification categories. To calculate the conditional probability given class APC, we classify each exemplar of APC against the model. Suppose we have 200 exemplars of class APC, and the numbers of exemplars giving result of 1 to the three categories are

170, 20, and 10, respectively. Then by taking the average, the conditional probability  $P(\text{TANK-VEHICLE} \mid \text{APC}) = 170 / 200 = 0.85$ ,  $P(\text{AIR-DEFENSE-GUN} \mid \text{APC}) = 0.1$ , and  $P(\text{SAUDI-NAVAL-MISSILE-CRAFT} \mid \text{APC}) = 0.05$ .

## 4 Experiments and Results

A large number of ontologies of different sizes have been used in many of our experiments. Due to the page limit, we only report some representative experiments, which involved two sets of ontologies. The first involves a small ontology whose structure is shown in Figure 3. We performed text classification between classes within this ontology and also with some foreign classes. The second set, WeaponsA.n3 and WeaponsB.n3, were taken from I<sup>3</sup>Con2004 [14].

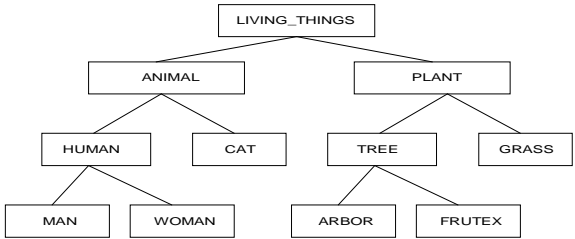


Fig. 3. Structure of LIVING\_THING ontology

The system was implemented on Linux and different components developed in perl or Java are glued together by shell scripts. The whole process from parsing, generating queries, to collecting exemplars, building models and calculating results is fully automated.

### 4.1 Results for Weapons Ontologies

Usually to generate a query for a class, the parser will use all the classes along the path from the root to the class in question. For weapons ontologies, because of their high specificity, to insure that sufficient web pages are returned, we decided to let the parser generate shorter queries, using only the class itself and its parent class.

Onto<sub>A</sub>, WeaponsA.n3 has more than 60 leaf classes. The model builder ran in default mode, and built a model using these leaf classes as classification categories. The retriever collected 100 exemplars on average for each class. The processor was invoked in two different ways and the results were compared. One is the default mode in which the entire text body of a web page is retained as a pure text exemplar; the other is to only keep sentences containing any of the search keywords as exemplars.

There are 9 classes in WeaponsB.n3 that do not appear in WeaponsA.n3. We try to find a mapping for each of them in WeaponsA.n3. These 9 classes and the manually selected desired mapping leaf classes in WeaponsA.n3 are listed in Table 1.

The conditional probabilities obtained are given in Table 2. For space limitation, here we only list the classes that have the highest probability instead of the complete results over 60 leaf classes for each of the 9 classes. The first column contains classes from WeaponsB.n3. The second and the third columns are the classes in WeaponsA.n3 with the highest conditional probability obtained by using a whole file as an exemplar. The last two columns are results obtained by using only sentences containing keywords as an exemplar.

If we simply judge the mapping accuracy by looking at the class that has the highest conditional probability, it is easy to see that when a whole text document is used as an exemplar, the accuracy is 11% (only LIGHT-AIRCRAFT-CARRIER is correctly mapped). However, the results are improved significantly if we use sentences containing keywords, as exemplars. The accuracy is 56% in this case. There

**Table 1.** Classes and their desired mappings

Classes from WeaponsB.n3	Desired leaf class mappings
LIGHT-AIRCRAFT-CARRIER	AIRCRAFT-CARRIER
APC	TANK-VEHICLE
SUPER-ETENDARD-FIGHTER	SUPER-ETENDARD
FIGHTER-ATTACK-PLANE	SUPER-ETENDARD
PATROL-WATERCRAFT	PATROL-CRAFT
PATROL-BOAT-RIVER	PATROL-CRAFT
PATROL-BOAT	PATROL-CRAFT
LIGHT-TANK	TANK-VEHICLE
FIGHTER-PLANE	SUPER-ETENDARD

**Table 2.** Classes with highest conditional probability

New Classes	Whole file	Prob	Keywords Sentences	Prob
LIGHT-AIRCRAFT-CARRIER	AIRCRAFT-CARRIER	0.65	AIRCRAFT-ARRIER	0.57
APC	SILKWORM-MISSILE-MOD	0.46	SELF-PROPELLED-RTILLERY	0.36
SUPER-ETENDARD-FIGHTER	SILKWORM-MISSILE-MOD	0.66	(BALLISTIC-MISSILE) RBM	0.51
FIGHTER-ATTACK-PLANE	SILKWORM-MISSILE-MOD	0.83	(BALLISTIC-MISSILE) RBM	0.38
PATROL-WATERCRAFT	SILKWORM-MISSILE-MOD	0.28	PATROL-CRAFT	0.52
PATROL-BOAT-RIVER	SILKWORM-MISSILE-MOD	0.65	PATROL-CRAFT	0.54
PATROL-BOAT	SILKWORM-MISSILE-MOD	0.51	PATROL-CRAFT	0.66
LIGHT-TANK	SILKWORM-MISSILE-MOD	0.56	TANK-VEHICLE	0.3
FIGHTER-PLANE	AIRCRAFT-CARRIER	0.49	(BALLISTIC-MISSILE) RBM	0.38

are four classes, APC, FIGHTER-PLANE, FIGHTER-ATTACK-PLANE, and SUPER-ETENDARD-FIGHTER, whose desired mapping classes do not have the highest conditional probability. We can see that by keeping only sentences containing keywords in an exemplar, noisy information in some web pages can be filtered out, which results in a better classification.

We further looked into those classes that did not get a correct mapping. For class APC, its desired mapping class TANK-VEHICLE has the second highest conditional probability (0.28). We also notice that the one with the highest conditional probability, SELF-PROPELLED-ARTILLERY is also closely related to APC (Armored Personnel Carrier). Text classification method and conditional probability can tell how related two concepts are, but the fact that two concepts are closely related does not mean that they are identical or similar semantically. This case is an example of an interesting problem for future research. For class SUPER-ETENDARD-FIGHTER, its desired mapping class SUPER-ETENDARD also has the second highest conditional probability (0.21). For the other two FIGHTER classes, the results are not good at all. We think one reason is SUPER-ETENDARD is the only leaf node in WeaponsA.n3 that represents a plane (violation of exhaustive assumption for categories). It is possible that it is indeed not a perfect mapping for some plane classes from WeaponsB.n3. To test this, we added a class WARPLANE-OTHER under the class WARPLANE in WeaponsA.n3, containing exemplars retrieved with a search query “WARPLANE+SUPER+ETENDARD” and performed the classification process again. Class FIGHTER-PLANE was mapped to its desired WARPLANE-OTHER with the highest conditional probability of 0.41. While class FIGHTER-ATTACK-PLANE still got a wrong mapping. This shows that adding a complement class helps. Moreover, FIGHTER-PLANE is a super class of the other two. The fact that a super class can be correctly mapped will make the mapping of its sub classes easier.

## 4.2 Results for Living\_things Ontology

To obtain further insights of this approach, we conducted the following additional experiments using the “living\_things” ontology shown in Fig.3.

1. Calculate  $P(\text{MAN} \mid \text{HUMAN})$  and  $P(\text{WOMAN} \mid \text{HUMAN})$ , both expected to be around 0.5.
2. Given a new, foreign class GIRL, build a model with classes ANIMAL and PLANT as the classification categories and perform classification. If class GIRL is mapped to class ANIMAL, then repeat this process by building a model with Class HUMAN and CAT, and so on.

The system performed these experiments automatically with at most 500 exemplars for each class. Extensive experiments were done with varying parameters. The typical results are reported in Table 3 and 4 (all using sentence-based exemplars).

What is disturbing is that Class CAT has a comparatively high conditional probability given GIRL. This was present during all the experiments we performed for this set of ontology. One reason for this anomaly is that words like human, man,

**Table 3.** Results of experiment 1

$P(\text{MAN} \mid \text{HUMAN})$	0.62
$P(\text{WOMAN} \mid \text{HUMAN})$	0.38

**Table 4.** Results of Experiment 2

$P(\text{ANIMAL} \mid \text{GIRL})$	0.76
$P(\text{PLANT} \mid \text{GIRL})$	0.23
$P(\text{HUMAN} \mid \text{GIRL})$	0.70
$P(\text{CAT} \mid \text{GIRL})$	0.30
$P(\text{MAN} \mid \text{GIRL})$	0
$P(\text{WOMAN} \mid \text{GIRL})$	1

woman and girl often appear in web pages associated with class CAT because cats have such close relations with human beings (sometimes cat is even used to describe a human). Manual inspection of the exemplars supports this reason.

The “cat” problem shows that even with the best parameters, the exemplars obtained with our system may still be far from perfect. This problem was further confirmed by an additional experiment in which DOG (another domesticated animal) and PYCNOGONID (a kind of sea spider) were added into the ontology as subclasses of ANIMAL. Most of the exemplars of GIRL went to Dog, and none to PYCNOGONID as shown in Table 5.

**Table 5.** Results with additional classes

$P(\text{DOG} \mid \text{GIRL})$	0.57
$P(\text{CAT} \mid \text{GIRL})$	0.03
$P(\text{HUMAN} \mid \text{GIRL})$	0.40
$P(\text{PYCNOGONID} \mid \text{GIRL})$	0

We conjecture that, although all exemplars for CAT taken as a whole are closely related to GIRL, it is different at the level of individual exemplars, some are close but others are not. The CAT problem can then be solved if we can separate exemplars that truly reflect the intended semantics of CAT from those that are not. As a first step, we have tried to perform clustering on exemplars of each class in the hope that one of the clusters would contain those truly relevant exemplars. We replaced Google with a clustering search engine Clusty.com that automatically clusters search results based on some text clustering algorithm. Then the largest cluster for each class returned by Clusty.com is used as exemplars. Results are a lot better as shown in Table 6.

We also tried to cluster exemplars obtained by Google search with clustering package in WEKA [15]. Taking the largest cluster does not yield good results this time. These limited experiments indicate that clustering of exemplars is promising in resolving the “CAT” problem, provided we find a way to identify the right clusters.

**Table 6.** Results by applying clustering on exemplars

P(ANIMAL   GIRL)	0.83
P(PLANT   GIRL)	0.17
P(HUMAN   GIRL)	0.92
P(CAT   GIRL)	0.08
P(WOMAN   GIRL)	0.63
P(MAN   GIRL)	0.37

## 5 Related Work

Many people have used text classification methods to solve the ontology mapping problem, they include, for example, OntoMapper [9], CAIMEN [7], and GLUE [8]. Although differing in technical details, one thing in common for these methods is that they all require the text exemplars for each concept class be given, and in their experiments, these exemplars are all *manually* collected. To our knowledge, no one has tried to automatically retrieve text exemplars from the web for this purpose.

On the other hand, some researchers in other applications do treat the WWW as a big sampling pool. For example, [16] also uses Google search results to estimate conditional probabilities. For example,  $P(\text{MAN} \mid \text{HUMAN})$  would be calculated as the ratio of the number of results for “man” and that for “human+man” (result changes as the number of pages found changes). This method depends on how likely MAN appears on web pages where HUMAN appears; while our method depends on the similarity of pages containing MAN, pages containing HUMAN and pages containing “NOT HUMAN”, which brings more semantic information in the contexts and ensures that the probabilities of all the leave classes sum to 1.

## 6 Conclusions

We proposed to automatically retrieve exemplars from the web for text classification based ontology mapping. We designed and implemented a fully automated system to collect exemplars and calculate conditional probability of two concepts as an initial similarity mapping. The tool can be very useful for ontology mapping tools and frameworks like [7, 8, 9, 12, and 13] and other researches using such a conditional probability [16].

Although our experiment results are mixed, they are in general encouraging and shed lights to the insight of this approach and further work. Two factors probably are most responsible for the less-than-ideal results. The first is the noise in the search results as revealed by the “cat” problem. This may be because that many search results are not really semantically relevant to the keywords, or they are relevant but not semantically close to keywords. The second is that a search is not really a random sampling of the web because all search engines return results according to their own ranking algorithms. How to address these problems and how to best utilize the imperfect exemplars in ontology mapping are the directions for future research.

## References

1. Berners-Lee T.: The Semantic Web. *Scientific American*, 284(5), (2001) 35-35.
2. Wiesman F., Roos N.: Domain Independent Learning of Ontology Mappings. *Proc of AAMAS* (2004).
3. Ushold M., Menzel C.: Achieving Semantic Interoperability & Integration Using RDF and OWL. <http://cmenzel.org/w3c/SemanticInterop.html>.
4. Reed S.L., Lenat D.: Mapping Ontologies into Cyc. *Proc of AAAI* (2002).
5. Niles I., Pease A.: Mapping WordNet to the SUMO Ontology, *Proc of the IEEE International Knowledge Engineering conference* (2003).
6. Li J.: LOM a lexicon based ontology mapping tool. <http://reliant.tekknowledge.com/DAML/I3con.pdf>.
7. Lacher M., Groh G.: Facilitating the Exchange of Explicit Knowledge through Ontology Mappings. *Proc of the 14<sup>th</sup> International FLAIRS conference* (2001).
8. Doan A., Madhavan J. et al: Learning to Match Ontologies on the Semantic Web. *Proc. WWW2002* (2002).
9. Sushama P., Peng Y., Finin T.: A Tool for Mapping between Two Ontologies Using Explicit Information. *AAMAS 2002 Workshop on Ontologies and Agent Systems* (2002).
10. McCallum A.: Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow> (1996)
11. Mitchell T.: *Machine Learning*, McGraw Hill. (1997)
12. Ding Z., Peng Y., Pan R.: BayesOWL: Uncertainty Modeling in Semantic Web Ontologies. *Soft Computing in Ontologies and Semantic Web*. Springer-Verlag, December (2005)
13. Ding Z., Peng Y., Pan R., Yu Y.: A Bayesian Methodology towards Automatic Ontology Mapping. *Proc of AAAI C&O-2005 Workshop*. (2005)
14. <http://www.atl.imco.com/projects/ontology/i3con.html>.
15. <http://www.cs.waikato.ac.nz/~ml/>.
16. Perkowski, M., Philipose, M. et al: Mining models of human activities from the web. In *Proceedings of WWW-04*. (2004) 573–582.



# On Combining Dissimilarity-Based Classifiers to Solve the Small Sample Size Problem for Appearance-Based Face Recognition\*

Sang-Woon Kim<sup>1</sup> and Robert P.W. Duin<sup>2</sup>

<sup>1</sup> *Senior Member, IEEE.* Dept. of Computer Science and Engineering,  
Myongji University, Yongin, 449-728 South Korea  
`kimsw@mju.ac.kr`

<sup>2</sup> Faculty of Electrical Engineering, Mathematics and Computer Science,  
Delft University of Technology, The Netherlands  
`r.p.w.duin@tudelft.nl`

**Abstract.** For high-dimensional classification tasks, such as face recognition, the number of samples is smaller than the dimensionality of the samples. In such cases, a problem encountered in Linear Discriminant Analysis-based (LDA) methods for dimension reduction is what is known as the Small Sample Size (SSS) problem. A number of LDA-extension approaches that attempt to solve the SSS problem have been proposed in the literature. Recently, a different way of employing a dissimilarity representation method was proposed [18], where an object was represented based on the dissimilarity measures among representatives extracted from training samples instead of the feature vector itself. Apart from utilizing the dissimilarity representation, in this paper, a new way of employing a fusion technique in representing features as well as in designing classifiers is proposed in order to increase the classification accuracy. The proposed scheme is completely different from the conventional ones in terms of the computation of the transformation matrix as well as the selection of the number of dimensions. The present experimental results demonstrate that the proposed combining mechanism works well and achieves further improved efficiency compared with the LDA-extension approaches for well-known face databases involving AT&T and Yale databases. The results especially demonstrate that the highest accuracy rates are achieved when the combined representation is classified with the trained combiners.

## 1 Introduction

Over the past two decades, numerous families and avenues for Face Recognition (FR) systems have been developed. This development is motivated by the broad range of potential applications for such identification and verification techniques.

---

\* The work of the first author was partially done while visiting at Delft University of Technology, 2628CD Delft, The Netherlands. This work was generously supported by KOSEF, the Korea Science and Engineering Foundation (F01-2006-000-10008-0).

Recent surveys are found in the literature [1] and [2] related to FR. As facial images are very high-dimensional, it is necessary for FR systems to reduce these dimensions. Linear Discriminant Analysis (LDA) is one of the most popular linear projection techniques for dimension reduction [3]. The major limitation when applying LDA is that it may encounter what is known as the Small Sample Size (SSS) problem [4], [5]. This problem arises whenever the number of samples is smaller than the dimensionality of the samples. Under these circumstances, the sample scatter matrix can become singular, and the execution of LDA may encounter computation difficulties.

In order to address the SSS issue, numerous methods have been proposed in the literature. One popular approach that addresses the SSS problem is to introduce a Principal Component Analysis (PCA) step to remove the null space of the between- and within-class scatter matrices before invoking the LDA execution. However, recent research reveals that the discarded null space may contain the most significant discriminatory information. Moreover, other solutions that use the null space can also have problems. Due to insufficient training samples, it is very difficult to identify the true null eigenvalues. Since the development of the PCA+LDA [3], other methods have been proposed successively, such as the pseudo-inverse LDA [6], the regularized LDA [7], the direct LDA [8], the LDA/GSVD [5] and the LDA/QR [9]. In addition to these methods, the Discriminative Common Vector (DCV) technique [10], has recently been reported to be an extremely effective approach to dimension reduction problems. The details of these LDA-extension methods are omitted here as they are not directly related to the premise of the present work.

Recently, a new paradigm to pattern classification has been proposed [11] - [13] based on the idea that if “similar” objects can be grouped together to form a class, the “class” is nothing more than a set of these similar objects. This methodology is a way of defining classifiers between the classes. It is not based on the feature measurements of the individual patterns, but rather on a suitable dissimilarity measure between them. The advantage of this is clear: As it does not operate on the class-conditional distributions, the accuracy can exceed the Bayes’ error bound. Another salient advantage of such a paradigm is that it does not have to confront the problems associated with feature spaces such as the “curse of dimensionality”, and the issue of estimating large numbers of parameters. Particularly, by selecting a set of prototypes or support vectors, the problem of dimension reduction can be *drastically* simplified.

On the other hand, combination systems which fuse “pieces” of information have received considerable attention because of its potential to improve the performance of individual systems. Various fusion strategies have been proposed in the literature and workshops<sup>1</sup> - excellent studies are found in [14], [15], and [16]. The applications of these systems are many. For example, consider a design problem involving pattern classifiers. The basic strategy used in fusion is to solve the classification problem by designing a *set* of classifiers, and then combining the individual results obtained from these classifiers in some way to achieve

<sup>1</sup> <http://www.diee.unica.it/mcs/home.html>

reduced classification error rates. Therefore, the choice of an appropriate fusion method can further improve on the performance of the individual method. Various classifier fusion strategies have been proposed in the literature. The decision rules commonly used are *Product*, *Sum*, *Max*, *Min*, *Median*, and *Majority vote* rules. Their details can be found in [14] and [15].

Motivated by the methods mentioned above, a combined dissimilarity-based scheme is investigated to solve the SSS problem in FR.

Recently, Kim [18] experimented the utilization of the dissimilarity representation as a method for solving the SSS problem. Apart from utilizing the dissimilarity representation, in this paper, a new way of employing a fusion technique in representing features as well as in designing classifiers is proposed. The combined dissimilarity-based scheme is completely different from the conventional ones in terms of the computation of the transformation matrix and the selection of the number of dimensions. A problem that is encountered in this paper concerns solving the SSS problem when the number of available facial images per subject is insufficient. For this reason, *all* samples are initially represented with different dissimilarity measures<sup>2</sup> among the samples instead of the feature vectors themselves. However, in facial images there are many kinds of variations, such as pose, illumination, facial expression, and distance. To overcome this problem, an object is classified with a combined classifier designed in the dissimilarity space.

In some cases, newly generated features based on a certain feature combination could be more informative compared to the original features. To obtain more powerful representation, in this paper, the dissimilarity representations are first combined into new ones by building an extended matrix or by simply averaging them. Then, the object is classified by invoking a group of dissimilarity-based classifiers as the *base* classifiers designed in the newly created dissimilarity space. The final decision is obtained with a *fixed* or *trained* combiner which is applied to the outputs of the base classifiers. The details of these classifiers are included in the present paper. The present experimental results for well-known face databases demonstrate that the proposed combining mechanism works well and achieves further improved efficiency results compared with the conventional LDA-extension approaches.

Two modest contributions are claimed in this paper by the authors:

1. This paper lists the first reported results that reduce the dimensionality and solve the SSS problem by resorting to the combined dissimilarity-based classifiers. Although the result presented is only for a case when the task is face recognition, the proposed approach can also apply to other high-dimensional tasks, such as information retrieval and bioinformatics.
2. The paper contains a formal algorithm in which, to improve classification performances for high-dimensional tasks, a fusion strategy in representing features as well as in designing classifiers is employed. The paper also

---

<sup>2</sup> Here, dissimilarity representations are measured with Euclidean-based metrics, such as the Euclidean distance and the regional distance, with the intent of simplifying the problem. The details of these measures will be included in the present paper.

provides experimental results by which the rationale of the dissimilarity-based scheme for employing the fusion technique is proven to be valid.

To the best of the authors' knowledge, all of these contributions are novel to a field of high-dimensional classification such as image recognition. This paper is organized as follows: An overview is initially presented of the dissimilarity representation in Section 2. Following this, the algorithm that solves the SSS problem by incorporating the use of dissimilarity representation and a fusion strategy is presented. Experimental results for the real-life benchmark data sets are provided in Section 3, and the paper is concluded in Section 4.

## 2 Combining Dissimilarity-Based Classifiers (DBC's)

### 2.1 Foundations of DBC's

Let  $T = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \in \mathbb{R}^p$  be a set of  $n$  feature vectors in a  $p$ -dimensional space. Assume that  $T$  is a labeled data set so that  $T$  can be decomposed into, for example,  $c$  disjoint subsets  $\{T_1, \dots, T_c\}$  such that  $T = \bigcup_{k=1}^c T_k, T_i \cap T_j = \emptyset, \forall i \neq j$ . The goal is to design a DBC in an appropriate dissimilarity space constructed with this *training data* set and to classify an input sample  $\mathbf{z}$  into an appropriate class. To achieve this, first of all, a prototype set of class  $\omega_i$ ,  $Y_i = \{\mathbf{y}_1, \dots, \mathbf{y}_{m_i}\}, m = \sum_{i=1}^c m_i$ , is extracted from the training data,  $T_i$ .

Every DBC assumes the use of a dissimilarity measure,  $d$ , computed from the samples, where  $d(\mathbf{x}_i, \mathbf{y}_j)$  represents the dissimilarity between two samples,  $\mathbf{x}_i$  and  $\mathbf{y}_j$ . The dissimilarity computed between  $T$  and  $Y$  leads to a  $n \times m$  matrix,  $D(T, Y)$ , where  $\mathbf{x}_i \in T$  and  $\mathbf{y}_j \in Y$ . Consequently, an object  $\mathbf{x}_i$  is represented as a column vector as following:

$$(d(\mathbf{x}_i, \mathbf{y}_1), d(\mathbf{x}_i, \mathbf{y}_2), \dots, d(\mathbf{x}_i, \mathbf{y}_m))^T, 1 \leq i \leq n. \quad (1)$$

Here, the dissimilarity matrix  $D(\cdot, \cdot)$  is defined as a *dissimilarity space* on which the  $p$ -dimensional object,  $\mathbf{x}$ , given in the feature space, is represented as an  $m$ -dimensional vector  $d(\mathbf{x}, Y)$ , where if  $\mathbf{x} = \mathbf{x}_i$ ,  $d(\mathbf{x}_i, Y)$  is the  $i^{th}$  row of  $D$  matrix. In this paper, the column vector  $d(\mathbf{x}, Y)$  is simply denoted by  $d(\mathbf{x})$ , where the latter is an  $m$ -dimensional vector, while  $\mathbf{x}$  is  $p$ -dimensional.

From this perspective, it becomes clear that the dissimilarity representation can be considered as a *mapping* by which  $\mathbf{x}$  is translated into  $d(\mathbf{x})$ ; thus,  $m$  is selected as sufficiently small ( $m \ll p$ ), what is being worked in is essentially a space with much smaller dimensions. Based on this consideration, the mapping could be considered as a way of solving the SSS problem.

Two factors to consider for a dissimilarity representation are to select a prototype subset from the training samples and to quantify the dissimilarity between two vectors. To do these things, various representative selection methods and dissimilarity measures have been proposed in [12], [13], and [17]. The details of these are omitted here in the interest of compactness.

## 2.2 Classifier Fusion Strategies (CFSs)

Recently, classifier combination (“Fusion”) has received considerable attention because of its potential to improve the performance of classification systems. The basic idea is to solve each classification problem by designing a *set* of classifiers, and then combining the classifiers in some way to achieve reduced classification error rates. Therefore a choice of an appropriate fusion method can further improve on the performance of the combination. Various CFSs have been proposed in the literature - excellent studies are found in [14], [15], and [16]. The CFS’s decision rules of [15] are summarized here briefly.

Consider a pattern recognition problem where pattern  $z$  is to be assigned to one of the  $c$  possible classes,  $\omega_1, \dots, \omega_c$ . Assume that there are  $M$  classifiers each representing the given pattern by a distinct measurement vector. Denote the measurement vector used by the  $i$ th classifier by  $\mathbf{x}_i, i = 1, \dots, M$ . In this case, the Bayesian decision rule computes the *a posteriori* probability  $p(\omega_k | \mathbf{x}_1, \dots, \mathbf{x}_M)$  using the Bayes theorem as follow:

$$p(\omega_k | \mathbf{x}_1, \dots, \mathbf{x}_M) = \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_M | \omega_k) P(\omega_k)}{\sum_{j=1}^c p(\mathbf{x}_1, \dots, \mathbf{x}_M | \omega_j) P(\omega_j)}. \quad (2)$$

Let us assume that the representations used are statistically independent. Then the joint probability distribution of the measurements extracted by the classifiers can be rewritten as follow:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_M | \omega_k) P(\omega_k) = \prod_{i=1}^M p(\mathbf{x}_i | \omega_k), \quad (3)$$

where  $p(\mathbf{x}_i | \omega_k)$  is the measurement process model of the  $i$ th representation.

Based on (2) and (3), the commonly used decision rules, such as *Product*, *Sum*, *Max*, *Min*, *Median*, and *Majority vote* rules, are obtained. Their details can be found in [14] and [15]. Although all of them can be used in a CFS, a rule used in the present experiment, namely, the *Majority vote* rule which operates under the assumption of equal priors, can be described as follows:

$$\sum_{i=1}^M \Delta_{ji} = \max_{1 \leq k \leq c} \left\{ \sum_{i=1}^M \Delta_{ki} \right\} \Rightarrow z \in \omega_j, \quad (4)$$

$$\Delta_{ki} = \begin{cases} 1, & \text{if } p(\omega_k | \mathbf{x}_i) = \max_{1 \leq j \leq c} \{p(\omega_j | \mathbf{x}_i)\}. \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Here, for each class  $\omega_k$ , the sum of  $\Delta_{ji}$  simply counts the votes received for this result from the individual classifiers. Thus the class which receives the largest number of votes is then selected as the majority decision.

The above combination schemes can be applied for combining a *set* of distinct features as well as different classifiers. Here, it is interesting to note that a number of distinct dissimilarity representations can be combined into a new one to obtain a more powerful representation in the discrimination. The idea

of this *feature* combination is derived from the possibility that discriminative properties of different representations can be enhanced by a proper fusion [12]. There are several schemes for combining multiple representations to solve a given classification problem. Some of them are : *Average*, *Product*, *Min*, and *Max* rules. The details of these methods are omitted here, but can be found in [12].

The reasons for combining several distinct dissimilarity representations and different dissimilarity-based classifiers will be exhaustively investigated in the present paper.

### 2.3 Combined Dissimilarity-Based Classifiers (CDBC's)

In this section, a dissimilarity-based method of classifying the high-dimensional samples without encountering the SSS problem is proposed. A simple Dissimilarity-Based Classifier (DBC) [17] consists of the following steps:

1. Select the representative set,  $Y$ , from the training set  $T$  by resorting to one of the prototype selection methods as described in [13], [17].
2. Compute the dissimilarity matrix,  $D(T, Y)$ , with  $T$  and  $Y$ , in which each individual dissimilarity is computed using one of the measures. To test a sample  $\mathbf{z}$ , compute a dissimilarity column vector,  $d(\mathbf{z})$ , using the same measure.
3. Achieve a classification based on invoking a classifier built in the dissimilarity space and operating on the dissimilarity vector  $d(\mathbf{z})$ .

However, in facial images there are many kinds of variations based on such factors as pose, illumination, facial expression, and distance. Thus, by simply measuring the differences of facial images for each class, it is not possible to obtain a good representation. To overcome this limitation, a classifier fusion strategy is employed. The basic strategy used in fusion is to solve the classification problem by designing a set of classifiers, and then to combine the individual results obtained from these classifiers in some way to achieve reduced classification error rates. The tangible rationale for this fusion strategy will be presented in a later section together with the experimental results.

The proposed approach, which is referred to as a Combined Dissimilarity-Based Classifier (CDBC), is summarized in the following:

1. Select the input training data set  $T$  as a representative subset  $Y$ .<sup>3</sup>
2. Compute dissimilarity matrices,  $D^{(1)}(T, Y)$ ,  $D^{(2)}(T, Y)$ ,  $\dots$ ,  $D^{(k)}(T, Y)$ , by using the  $k$  different dissimilarity measures for all  $\mathbf{x} \in T$  and  $\mathbf{y} \in Y$ .
3. To obtain more powerful representation, combine the dissimilarity matrices,  $\{D^{(i)}(T, Y)\}_{i=1}^k$ , into new ones,  $\{D^{(j)}(T, Y)\}_{j=1}^l$ , by building an extended matrix or by computing their weighted average.

---

<sup>3</sup> This is a *Wholeset* method. Undoubtedly, for “large size” applications, we can select the small number of representatives from the given training data set through the clustering phase. Rather than deciding to discard or retain the training points with the *Random\_C*, *PeatSeal*, or *KCentres* [13], we can do this by invoking a PRS (Prototype Reduction Scheme). For the interest of brevity, the details of the *PRS-based methods* are omitted here, but can be found in [17].

4. For any dissimilarity matrix,  $D^{(j)}(T, Y)$ , ( $j = 1, \dots, l$ ), perform classification of the input,  $\mathbf{z}$ , with *combined* classifiers designed on the newly created dissimilarity space as follows:
  - (a) Compute a dissimilarity column vector,  $d^{(j)}(\mathbf{z})$ , for the input sample  $\mathbf{z}$ , with the same method as in measuring the  $D^{(j)}(T, Y)$ .
  - (b) Classify  $d^{(j)}(\mathbf{z})$  by invoking a group of DBCs as the *base* classifiers designed with  $n$   $m$ -dimensional vectors in the dissimilarity space. The classification results are labeled as  $class_1, class_2, \dots, class_M$ , respectively.
5. Obtain the final result from the  $class_1, class_2, \dots, class_M$  by combining the base classifiers designed in the above step, where the base classifiers are combined to form the final decision in the *fixed* or *trained* fashion.

In the above algorithm, using the  $n \times n$  dissimilarity matrix, the feature-based vectors are translated into the *dissimilarity-based vectors*, where the dimensionality is determined with the number of samples  $n$ . While the dimensionality of the feature-based vectors is  $p$ , thus, the dimensionality of the dissimilarity-based vectors is  $n (< p)$ . Notice also that the sample to be tested is projected onto the dissimilarity space represented by the dissimilarity matrix. From these considerations, it can be noted that the algorithm can be used as a scheme to reduce the dimensionality without encountering the SSS problem in FR.

In Step 3, on the other hand, a number of distinct dissimilarity matrices can be combined into a new one to obtain a more powerful representation in the discrimination. A simple method to do this is to average different representations. For example, two dissimilarity matrices,  $D^{(1)}(T, Y)$  and  $D^{(2)}(T, Y)$ , can be averaged into  $\frac{1}{2}(\alpha_1 D^{(1)}(T, Y) + \alpha_2 D^{(2)}(T, Y))$  after scaling with an appropriate weight,  $\alpha_\tau$ , to guarantee that they all take values in a similar range. In addition to this averaging method, the two dissimilarity matrices can be combined into:  $\sum_{\tau=1}^2 \log(1 + \alpha_\tau D^{(\tau)}(T, Y))$ ,  $\min_\tau \{\alpha_\tau D^{(\tau)}(T, Y)\}$ , and  $\max_\tau \{\alpha_\tau D^{(\tau)}(T, Y)\}$  [12]. Some of them will be exhaustively investigated in the present experiment.

The computational complexity of the proposed algorithm depends on the computational costs associated with the dissimilarity matrix. The time complexity of CDBC can be analyzed as follows: Step 1 requires  $O(1)$  time. Step 2 requires  $k \times O(n^2) = O(n^2)$  time to compute the  $k$  dissimilarity matrices. Step 3 requires  $l \times O(n^2) = O(n^2)$  time to compute the  $l$  combined matrices, for example, by averaging the  $l$  matrices. Step 4 requires  $O(n) + M \times O(\gamma_1) = O(\gamma_1)$  time (where  $M$  is the number of the base classifiers and  $\gamma_1$  is the time for doing classification with the base classifiers.) to project the test sample onto the dissimilarity space and classify it with the *base* classifiers designed in the dissimilarity space. Step 5 requires  $O(\gamma_2)$  time to classify the test sample with the *combined* classifier designed in the dissimilarity space. Here,  $\gamma_2$  is the time for obtaining the final result. Thus, the total time complexity of the CDBC is  $O(n^2 + \gamma_1 + \gamma_2)$ . Then, the space complexity of CDBC is  $O(n(n + p))$ .<sup>4</sup>

<sup>4</sup> In [9], it was reported that the time complexities of LDA-extension methods such as PCA, PCA+LDA, LDA/GSVD, and RLDA, respectively, are  $O(n^2p)$ ,  $O(n^2p)$ ,  $O((n + c)^2p)$ , and  $O(n^2p)$  and their space complexities are all the same as  $O(np)$ .



### 3 Experimental Results

#### 3.1 Experimental Data

The proposed method has been tested and compared with conventional methods. This was done by performing experiments on two well-known benchmark face databases, namely, the “AT&T”<sup>5</sup> and “Yale”<sup>6</sup> databases.<sup>7</sup>

The face database captioned AT&T, formerly the ORL database of faces, consists of ten different images of 40 distinct subjects, for a total of 400 images. Each subject is positioned upright in front of a dark homogeneous background. The size of each image is  $112 \times 92$  pixels, for a total dimensionality of 10304. The face database termed as Yale contains 165 gray scale images of 15 individuals. The size of each image is  $243 \times 320$  pixels, for a total dimensionality of 77760. However, in this experiment, each facial image of  $236 \times 178$  pixels was manually extracted, and then represented by a centered vector of normalized intensity values.

#### 3.2 Experimental Method

In this paper, all experiments were performed using a “leave-one-out” strategy. To classify an image of object, that image is removed from the training set and the dissimilarity matrix is computed with the  $n - 1$  images. Following this, all of the  $n$  images in the training set and the test object were translated into a dissimilarity space using the dissimilarity matrix, and recognition was performed based on the proposed algorithm in Section 2.3. We repeated this  $n$  times for every sample and obtained a final result by averaging them.

To construct the dissimilarity matrix, all samples were selected as representatives and the dissimilarities were measured with the Euclidean distance and the regional distance. Here the two distance measures are named as “ED” and “RD”, respectively.<sup>8</sup> The distance measure called RD is defined as the average of the minimum difference between the gray value of a pixel and the gray value of each pixel in the  $5 \times 5$  neighborhood of the corresponding pixel. In this case, the regional distance compensates for a displacement of up to three pixels of the images. For the interest of brevity, the details of the distance measure are omitted here, but can be found in the literature including [19].

However, the faces for some subjects vary with pose, illumination, facial expression, and whether or not they are wearing glasses. Thus, the dissimilarity

<sup>5</sup> <http://www.cl.cam.ac.uk/Research/DTG/attarchive/facedatabase.html>

<sup>6</sup> <http://www1.cs.columbia.edu/belhumeur/pub/images/yalefaces>

<sup>7</sup> A thorough evaluation on AT&T and Yale databases is presented here. It would be interesting to see results on more challenging datasets, such as FERET and CMU-PIE. The results on these datasets will appear in the next paper.

<sup>8</sup> Here, we experimented with two simple measures, namely, ED and RD. However, it should be mentioned that we can have numerous solutions, depending on dissimilarity measures, such as the Hamming distance, the modified Hausdorff distances, the blurred Euclidean distance, etc. From this perspective, the question “what is the best measure?” is an interesting issue for further study.



matrix simply obtained by measuring the input images can not work as a representative. To overcome this problem as well as the SSS problem, a combined dissimilarity representation and two classifier fusion strategies are employed in the experiment. To investigate this combination rule, first of all, two dissimilarity representations, namely, ED and RD, are averaged into a new representation (which is named as “AD” here) after normalization. As mentioned in the previous section, *three* base classifiers are designed in this newly defined dissimilarity space, and then all of their results are combined in *fixed* or *trained* fashion.

Since the diversity between the base classifiers is essential for constructing a robust ensemble, different classifiers, such as Nearest Mean Classifiers, Normal Density based Classifiers, and Nonlinear Classifiers, are considered as the base classifiers. These three kinds of base classifiers are implemented with PRTools,<sup>9</sup> and will be denoted as *nmc*, *ldc*, and *knnc*, respectively, in a subsequent section. The outputs of the base classifiers are combined with fixed combiners, such as *Product*, *Median*, and *Majority vote* rules, and two trained classifiers. All *five* combiners are also implemented with PRTools, and named as *prodc*, *medianc*, *votec*, *meanc*, and *fisherc*, respectively. To simplify the classification task for the paper, only three base classifiers, three fixed and two trained combiners are experimented. However, other classifiers, including neural network and SVM based classifiers, and combining rules can also be considered.

### 3.3 Experimental Results

The run-time characteristics of the proposed algorithm for the two benchmark databases, AT&T and Yale, is reported below and shown in Table 1. The performance of the dissimilarity-based classifiers (DBC and CDBC) is investigated first. Following this, a comparison is made between the conventional LDA-extension methods and the proposed CDBC scheme.

First of all, to examine the rationality of employing a fusion technique in the CDBC, the simple Dissimilarity-Based Classifier (DBC) was experimented. While CDBC involves all of the five steps given in Section 2.3, DBC consists of only the steps 1, 2, and 4 with  $k = 1$  and  $l = 1$ . The classification accuracy rates of DBC was evaluated for the AT&T and Yale databases. In this experiment, the same dissimilarity matrix was constructed for both DBC and CDBC.

Table 1 shows the classification accuracy rates (%) of DBCs and CDBC for the two databases. Here, the abbreviations ED, RD, and AD, which are the Euclidian distance, the regional distance, and the averaged distance, indicate the dissimilarity measures employed in this experiment. Additionally, in the base classifiers column, an Uncorrelated Normal based Quadratic Classifier (named as *udc*) was used for the RD representation instead of the Normal Density based Classifier (*ldc*). Also, *knnc* stands for the  $k$ -Nearest Neighbor Classifier ( $k = 1$ ).

From Table 1, it is observed that the classification accuracies for the benchmark databases can be improved by employing the philosophy of CDBC. This is clearly shown in the classification accuracy rates of the classifiers designed for

<sup>9</sup> PRTools is a Matlab Toolbox for Pattern Recognition. PRTools can be downloaded from the PRTools website, <http://www.prtools.org/>

**Table 1.** A comparison of classification accuracy rates (%) of the base Dissimilarity-Based Classifiers (DBC) and the Combined Dissimilarity-Based Classifiers (CDBC) designed with the fixed and the trained combiners. Here, the classifiers of *nmc*, *ldc* (*udc\**), and *knnc* are designed and evaluated as DBCs. Then, the combiners of *prodc*, *medianc*, and *votec* are employed as the fixed combining schemes of the DBCs. Finally, the classifiers of *meanc* and *fisherc* are employed as the trained combiners respectively.

Data Sets	Distance Measures	Base Classifiers			Fixed Combiners			Trained Combiners	
		<i>nmc</i>	<i>ldc(udc*)</i>	<i>knnc</i>	<i>prodc</i>	<i>medianc</i>	<i>votec</i>	<i>meanc</i>	<i>fisherc</i>
AT&T	ED	81.25	98.75	96.50	98.75	98.25	98.00	98.75	99.00
	RD*	71.25	88.00	95.00	88.00	89.25	89.00	88.00	89.00
	AD	76.25	99.25	95.75	99.25	98.50	98.00	99.25	99.25
Yale	ED	80.61	93.33	79.39	93.33	86.06	86.06	93.33	93.33
	RD*	78.18	72.12	79.39	72.12	76.36	76.97	72.12	78.18
	AD	79.39	96.36	78.79	95.76	82.42	86.67	96.36	96.36

the AT&T database measured with ED. Specifically, the classification accuracies of the base classifiers, namely, *nmc*, *ldc*, and *knnc*, are 81.25, 98.75, and 96.50 (%), while those of the fixed combiners, such as *prodc*, *medianc*, and *votec*, are 98.75, 98.25, and 98.00 (%), respectively. Additionally, the trained combiners of *meanc* and *fisherc* have the classification accuracies of 98.75 and 99.00 (%), respectively. From this consideration, it is evident that the rationale of the paper for employing a fusion technique works well. Furthermore, the result of the comparison is completely in accord with the well-known fact that the combination of different classifiers for the same feature set only slightly improves the best individual results. Besides this, the results also prove that the best overall result is obtained by a trained combiner. This is the case of the *fisherc* here. For the Yale database, the same characteristics can be observed.

Secondly, as the main results, it should be noted that it is possible to improve the classification performance by appropriately combining the dissimilarity representations. For instance, the classification accuracy rates of the three base classifiers designed with AD for AT&T database are (76.25, 99.25, 95.75) (%), respectively, and those of the three fixed and the two trained combiners applied to the outputs of the base classifiers are (99.25, 98.50, 98.00) and (99.25, 99.25) (%), respectively. The above comparison shows that the accuracy rates of the combiners are generally higher than those of the base classifiers. From these considerations, the reader should observe that the newly created dissimilarity representation of *AD* improves the performance of the classification accuracy more effectively than the ED or RD measure. Therefore, it can be concluded that the highest accuracy rates are achieved when the combined representation, namely, AD, is classified with the trained combiners. However, it should be also pointed out that the classification efficiencies were not improved in both combiners for RD. For the Yale database, the same characteristics can be observed.

Finally, CDBC can be compared with LDA-extensions for solving the SSS problem in FR. Consider experimental results on the LDA-extensions, such as the PCA [3], the PCA+LDA [3], the direct LDA [8], the DCV [10], and the

LDA/GSVD [5], which have been recently reported in [18]. In that experiment, to reduce the computational complexity, each facial image from the two databases, AT&T and Yale, was down-sampled into  $56 \times 46$  and  $61 \times 80$ , respectively. Additionally, the “leave-one-out” strategy was also used to experiment with these methods. In [18], the classification accuracies of the PCA, PCA+LDA, direct LDA, DCV, and LDA/GSVD methods for AT&T and Yale databases are (93.25, 95.50, 98.50, 97.25, 93.50) and (72.73, 74.55, 92.12, 70.91, 98.79) (%), respectively. A comparison of these figures and Table 1 shows that the classification accuracy of CDBC is marginally higher than that of the conventional methods. From this consideration, the rationale of the dissimilarity-based scheme for employing a fusion technique is proven to be valid.

In review, it is not easy to say that one specific method is superior to others for solving the SSS problem in FR. However, as a matter of comparison, it is clear that the combined dissimilarity-based method is better than the conventional schemes with regard to the classification accuracy rates.

## 4 Conclusions

In this paper a method that seeks to address the SSS problem of image recognition by combining the dissimilarity-based classifiers was considered. Rather than use Fisher’s criterion to reduce the dimensionality, a completely different approach was employed, in which an object was represented based on the dissimilarity measures among training samples instead of the feature vector itself. Apart from utilizing the dissimilarity representation [18], to increase the classification accuracy, in this paper, a new method of employing a fusion technique in representing features as well as in designing classifiers was proposed.

The proposed method has been tested on two well-known face databases and compared with LDA-extension approaches. The experimental results demonstrate that the proposed scheme works well and its classification accuracy is better than that of the conventional ones. The results especially demonstrate that the highest accuracy rates are achieved when the combined representation is classified with the trained combiners. Although an investigation was made that focused on the possibility that the combined dissimilarity-based classifiers could be used to solve the SSS problem, many problems remain. One of them is an improvement of the classification performance by utilizing an appropriate dissimilarity measure (i.e., a modified Hausdorff distance) and by developing a suitable feature combination in the dissimilarity space. The research concerning this is a future aim of the authors.

## References

1. W. Zhao, R. Chellappa, A. Rosenfeld, and P. J. Phillips, “Face recognition: a literature survey”, *ACM Comput. Surveys*, vol. 35, no. 4, pp. 399 - 458, 2003.
2. J. Ruiz-del-Solar and P. Navarrete, “Eigenspace-based face recognition: a comparative study of different approaches”, *IEEE Trans. Systems, Man, and Cybernetics - Part C*, vol. 35, no. 3, pp. 315 - 325, Aug. 2005.

3. P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection", *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 19, no. 7, pp. 711 - 720, July 1997.
4. L. -F. Chen, H. -Y. M. Liao, M. -T. Ko, J. -C. Lin, and G. -J. Yu, "A new LDA-based face recognition system which can solve the small sample size problem", *Pattern Recognition*, vol. 33, pp. 1713 - 1726, 2000.
5. P. Howland, J. Wang, and H. Park, "Solving the small sample size problem in face recognition using generalized discriminant analysis", *Pattern Recognition*, vol. 39, pp. 277 - 287, 2006.
6. S. Raudys and R. P. W. Duin, "On expected classification error of the Fisher linear classifier with pseudoinverse covariance matrix", *Pattern Recognition Letters*, vol. 19, pp. 385 - 392, 1998.
7. D. Q. Dai and P. C. Yuen, "Regularized discriminant analysis and its application to face recognition", *Pattern Recognition*, vol. 36, pp. 845 - 847, 2003.
8. H. Yu and J. Yang, "A direct LDA algorithm for high-dimensional data - with application to face recognition", *Pattern Recognition*, vol. 34, pp. 2067 - 2070, 2001.
9. J. Ye and Q. Li, "A two-stage linear discriminant analysis via QR-decomposition", *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 27, no. 6, pp. 929 - 941, Jun. 2005.
10. H. Cevikalp, M. Neamtu, M. Wilkes, and A. Barkana, "Discriminative common vectors for face recognition", *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 27, no. 1, pp. 4 - 13, Jan. 2005.
11. E. Pekalska and R. P. W. Duin, "Dissimilarity representations allow for building good classifiers", *Pattern Recognition Letters*, vol. 23, pp. 943 - 956, 2002.
12. E. Pekalska, *Dissimilarity representations in pattern recognition. Concepts, theory and applications*, Ph.D. thesis, Delft University of Technology, The Netherlands, 2005.
13. E. Pekalska, R. P. W. Duin, and P. Paclik, "Prototype selection for dissimilarity-based classifiers", *Pattern Recognition*, vol. 39, pp. 189 - 208, 2006.
14. L. I. Kuncheva, "A theoretical study on six classifier fusion strategies", *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 24, no. 2, pp. 281 - 286, Feb. 2002.
15. J. Kittler, M. Hatef, R. P. W. Duin and J. Matas, "On combining classifiers", *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 20, no. 3, pp. 226 - 239, Mar. 1998.
16. L. I. Kuncheva, J. C. Bezdek and R. P. W. Duin "Decision templates for multiple classifier fusion: an experimental comparison", *Pattern Recognition*, vol. 34, pp. 299 - 414, Feb. 2001.
17. S. -W. Kim and B. J. Oommen, "On optimizing dissimilarity-based classification using prototype reduction schemes", *The 2006 International Conference on Image Analysis and Recognition (ICIAR2006)*, LNCS 4141, pp. 15-28, 2006.
18. S. -W. Kim, "On solving the small sample size problem using a dissimilarity representation for face recognition", *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACTVS2006)*, LNCS 4179, pp. 1174-1185, 2006.
19. Y. Adini, Y. Moses, and S. Ullman, "Face recognition: The problem of compensating for changes in illumination direction", *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 19, no. 7, pp. 721 - 732, Jul. 1997.

# A Novel Approach for Automatic Palmprint Recognition

Murat Ekinici and Murat Aykut

Computer Vision Lab.  
Department of Computer Engineering,  
Karadeniz Technical University, Trabzon, Turkey  
ekinici@ktu.edu.tr

**Abstract.** In this paper, we propose an efficient palmprint recognition scheme which has two features: 1) representation of palm images by two dimensional (2-D) wavelet subband coefficients and 2) recognition by a modular, personalized classification method based on Kernel Principal Component Analysis (Kernel PCA). Wavelet subband coefficients can effectively capture substantial palm features while keeping computational complexity low. We then kernel transforms to each possible training palm samples and then mapped the high-dimensional feature space back to input space. Weighted Euclidean linear distance based nearest neighbor classifier is finally employed for recognition. We carried out extensive experiments on PolyU Palmprint database includes 7752 palms from 386 different palms. Detailed comparisons with earlier published results are provided and our proposed method offers better recognition accuracy (99.654%).

## 1 Introduction

Biometrics is becoming more and more popular in an increasingly automated world. Palmprint recognition is one kind of biometric technology and a relatively new biometric feature. Compared with other biometrics, the palmprints has several advantages: low-resolution imaging can be employed; low-cost capture devices can be used; it is difficult to fake a palmprint; the line features of the palmprints are stable, etc. [1]. It is for these reasons that palmprint recognition has recently attracted an increasing amount of attention from researchers.

There are many approaches for palmprint recognition using line-based [2][4][5], texture-based [9][5], and appearance-based methods [3][8][7][6] in various literature. In the line-based approach, the features used such as principal lines, wrinkles, delta points, minutiae, etc. are sometimes difficult to extract directly from a given palmprint image with low resolution. The recognition rates and computational efficiency are not strong enough for palmprint recognition. In the texture-based approach, the texture features are not sufficient and the extracted features are greatly affected by the lighting conditions. From that disadvantages, researches have developed the appearance-based approaches. The appearance-based approaches only use a small quantity of samples in each palmprint class

randomly selected as training samples to extract the appearance features (commonly called algebraic features) of palmprints and form feature vector.

Eigenpalms method [8], fisherpalms method [3], and eigen-and-fisher palms [7] are presented as the appearance-based approaches for palmprint recognition in literature. Basically, their representations only encode second-order statistics, namely, the variance and the covariance. As these second order statistics provide only partial information on the statistics both natural images and palm images, it might become necessary to incorporate higher order statistics as well. In other words, they are not sensitive to higher order statistics of features. A kernel fisherpalm [6] is presented as another work to resolve that problem. In addition, for palmprint recognition, the pixelwise covariance among the pixels may not be sufficient for recognition. The appearance of a palm image is also severely affected by illumination conditions that hinder the automatic palmprint recognition process.

Converging evidence in neurophysiology and psychology is consistent with the notion that the visual system analyses input at several spatial resolution scales [19]. Thus, spatial frequency preprocessing of palms is justified by what is known about early visual processing. By spatial frequency analysis, an image is represented as a weighted combination of basis functions, in which high frequencies carry finely detailed information and low frequencies carry coarse, shape-based information. Recently, there have been renewed interests in applying discrete transform techniques to solve some problems in face recognition [13][14][17], in palmprint recognition [17][18] and many real world problems. An appropriate wavelet transform can result in robust representations with regard to lighting changes and be capable of capturing substantial palm features while keeping computational complexity low.

From these all considerations, we propose to use discrete wavelet transform (DWT) to decompose palm images and choose the lowest resolution subband coefficients for palm representation. We then apply kernel PCA as a nonlinear method to project palmprints from the high-dimensional palmprint space to a significantly lower-dimensional feature space, in which the palmprints from the different palms can be discriminated much more efficiently. The main contributions and novelties of the current paper are summarized as follows:

- To reliably extract palmprint representation, we adopt a template matching approach where the feature vector of a palm image is obtained through a multilevel two-dimensional discrete wavelet transform (DWT). The dimensionality of a palm image is greatly reduced to produce the *waveletpalm*.
- A nonlinear machine learning method, kernel PCA, is applied to extract palmprint features from the *waveletpalm*.
- The proposed algorithm is tested on a public palmprint databases. We provide some quantitative comparative experiments to examine the performance of the proposed algorithm and different combinations of the proposed algorithm. Comparison between the proposed algorithm and other recent approaches is also given.

This paper is organized as follows. Section 2 introduces briefly wavelet transform, lowest subband image representation, and fast Fourier transform (FFT) which is also implemented in this work to compare the efficiencies on the palmprint recognition. A brief description of kernel PCA (KPCA) and similarity measurement used are given in Sections 3 and 4 respectively. Experimental results on the palmprint database are summarized in Section 5 followed by discussions and conclusions in Section 6.

## 2 Discrete Transforms

In the proposed algorithm, the palmprint is first transformed into the wavelet domain, then kernel PCA is applied to extract higher order relations among waveletpalms for future recognition. In order to compare the efficiencies of the wavelet transform and discrete fast Fourier transform (FFT) is alternately employed in the proposed algorithm.

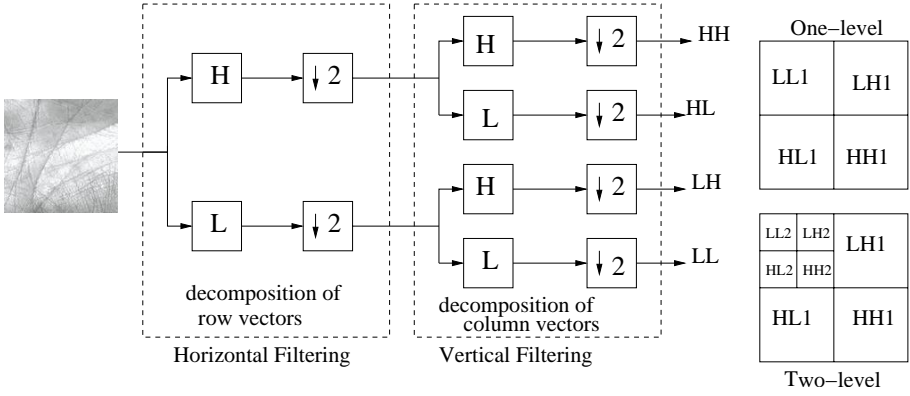
### 2.1 Discrete Wavelet Transform

The DWT was applied for different applications given in the literature e.g. texture classification [12], image compression, face recognition [13][14], because of its powerful capability for multiresolution decomposition analysis. The wavelet transform breaks an image down into four subsampled, or decimated, images. They are subsampled by keeping every other pixel. The results consist of one image that has been high pass filtered in both the horizontal and vertical directions, one that has been high pass filtered in the vertical and low pass filtered in the horizontal, one that has been lowpassed in the vertical and highpassed in the horizontal, and one that has been low pass filtered in both directions.

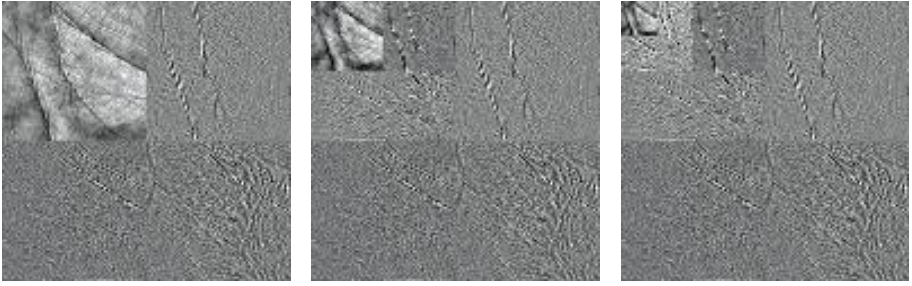
So, the wavelet transform is created by passing the image through a series of 2D filter bank stages. One stage is shown in Fig. 1, in which an image is first filtered in the horizontal direction. The filtered outputs are then down sampled by a factor of 2 in the horizontal direction. These signals are then each filtered by an identical filter pair in the vertical direction. Decomposed image into 4 subbands is also shown in Fig. 1. Here, H and L represent the high pass and low pass filters, respectively, and  $\downarrow 2$  denotes the subsampling by 2. Second-level decomposition can then be conducted on the LL subband. Second-level structure of wavelet decomposition of an image is also shown in Fig. 1. This decomposition can be repeated for n-levels. Fig. 2 shows one-level, two-level and three-level wavelet decomposition of a palm image.

The proposed work based DWT addresses the four-level decomposition of images in the database used for experiments. Daubechies-8 [11] low pass and high pass filters are also implemented. Additionally, four-level of decompositions are produced, then 32 x 32 sub-images of 128 x 128 images in the wavelet are processed as useful features in the palmprint images. Reduce of the image resolution helps to decrease the computation load of the feature extraction process.





**Fig. 1.** One-level 2-D filter bank for wavelet decomposition and multi-resolution structure of wavelet decomposition of an image



**Fig. 2.** Palm images with one-level, two-level, and three-level wavelet decomposition are shown

## 2.2 2-D Discrete FFT

$F(u, v)$  is 2-D FFT coefficients of  $W \times H$  image  $I(x, y)$ . The feature sequence is generated using the 2D-FFT technique. The palmprint image ( $128 \times 128$ ) in the spatial domain is not divided into any overlap blocks. The FFT coefficients for the palmprint image are first computed. In FFT, the coefficients correspond to the lower frequencies than  $3 \times 3$ , and to the higher frequencies than  $16 \times 16$  in FFT, were discarded by filtering. In other words, 247 coefficients ( $(16 \times 16) - (3 \times 3)$ ) correspond to the 6% coefficients in the frequency domain,  $(64 \times 64)$ , were only processed. These data are empirically determined to achieve the best performance. Therefore, the size of the palmprint image ( $128 \times 128$ ) in the spatial domain was reduced to the very few coefficients in the frequency domain correspond to the 1.5% coefficient. Finally,  $N = \mu x \nu$  features form a vector  $\chi \in \mathbb{R}^N$ ,  $\chi = (F_{0,0}, F_{0,1}, \dots, F_{\mu,\nu})$  for FFT.



### 3 Kernel PCA

The kernel PCA (KPCA) is a technique for nonlinear dimension reduction of data with an underlying nonlinear spatial structure. A key insight behind KPCA is to transform the input data into a higher-dimensional **feature space** [10]. The feature space is constructed such that a nonlinear operation can be applied in the input space by applying a linear operation in the feature space. Consequently, standard PCA can be applied in feature space to perform nonlinear PCA in the input space.

Let  $\chi_1, \chi_2, \dots, \chi_M \in \mathfrak{R}^N$  be the data in the input space (the input space is 2D-DWT coefficients in this work), and let  $\Phi$  be a nonlinear mapping between the input space and the feature space i.e. using a map  $\Phi : \mathfrak{R}^N \rightarrow F$ , and then performing a linear PCA in  $F$ . Note that, for kernel PCA, the nonlinear mapping,  $\Phi$ , usually defines a kernel function [10]. The most often used kernel functions are polynomial kernels, Gaussian kernels, and sigmoid kernels [10]:

$$k(\chi_i, \chi_j) = \langle \chi_i, \chi_j \rangle^d, \quad (1)$$

$$k(\chi_i, \chi_j) = \exp\left(-\frac{\|\chi_i - \chi_j\|^2}{2\sigma^2}\right), \quad (2)$$

$$k(\chi_i, \chi_j) = \tanh(\kappa \langle \chi_i, \chi_j \rangle + \vartheta), \quad (3)$$

where  $d$  is a number in the set of natural numbers, e.g.  $\{1, 2, \dots\}$ ,  $\sigma > 0$ ,  $\kappa > 0$ , and  $\vartheta < 0$ .

The mapped data is centered, i.e.  $\sum_{i=1}^M \Phi(\chi_i) = 0$  (for details see [10]), and let  $D$  represents the data matrix in the feature space:  $D = [\Phi(\chi_1) \Phi(\chi_2) \dots \Phi(\chi_M)]$ . Let  $K \in \mathfrak{R}^{M \times M}$  define a kernel matrix by means of dot product in the feature space:

$$K_{ij} = (\Phi(\chi_i) \cdot \Phi(\chi_j)). \quad (4)$$

The work in [10] shows that the eigenvalues,  $\lambda_1, \lambda_2, \dots, \lambda_M$ , and the eigenvectors,  $V_1, V_2, \dots, V_M$ , of kernel PCA can be derived by solving the following eigenvalue equation:

$$KA = MA\Lambda \quad (5)$$

with  $A = [\alpha_1, \alpha_2, \dots, \alpha_M]$  and  $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_M\}$ .  $A$  is  $MXM$  orthogonal eigenvector matrix,  $\Lambda$  is a diagonal eigenvalue matrix with diagonal elements in decreasing order ( $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M$ ), and  $M$  is a constant corresponds to the number of training samples. Since the eigenvalue equation is solved for  $\alpha$ 's instead of eigenvectors,  $V = [V_1, V_2 \dots V_M]$ , of kernel PCA, first,  $A$  should be normalized to ensure that eigenvalues of kernel PCA have unit norm in the feature space, therefore  $\lambda_i \|\alpha_i\|^2 = 1, i = 1, 2, \dots, M$ . After normalization the eigenvector matrix,  $V$ , of kernel PCA is then computed as follows:

$$V = DA \quad (6)$$

Now let  $\chi$  be a test sample whose map in the higher dimensional feature space is  $\Phi(\chi)$ . The kernel PCA features of  $\chi$  are derived as follows:

$$F = V^T \Phi(\chi) = A^T B \quad (7)$$

where  $B = [\Phi(\chi_1) \cdot \Phi(\chi) \Phi(\chi_2) \cdot \Phi(\chi) \dots \Phi(\chi_M) \Phi(\chi)]^T$ .

## 4 Similarity Measurement

When a palm image is presented to the wavelet-based kernel PCA classifier, the wavelet feature of the image is first calculated as detailed in Section 2, and the low-dimensional wavelet-based kernel PCA features,  $F$ , are derived using the equation 7. Let  $M_k^0, k = 1, 2, \dots, L$ , be the mean of the training samples for class  $w_k$ . The classifier applies, then, the nearest neighbor rule for classification using some similarity (distance) measure  $\delta$ :

$$\delta(F, M_k^0) = \min_j \delta(F, M_j^0) \longrightarrow F \in w_k, \quad (8)$$

The wavelet-based kernel PCA feature vector,  $F$ , is classified as belong to the class of the closest mean,  $M_k^0$ , using the similarity measure  $\delta$ .

Popular similarity measures include the Weighted Euclidean Distance (WED) and Linear Euclidean Distance (LED) which are defined as follows:

$$WED : d_k = \sum_{i=1}^N \frac{(f(i) - f_k(i))^2}{(s_k)^2} \quad (9)$$

where  $f$  is the feature vector of the unknown palmprint,  $f_k$  and  $s_k$  denote the  $k$ th feature vector and its standard deviation, and  $N$  is the feature length.

$$LED : d_{ij}(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0 \quad (10)$$

where  $d_{i,j}$  is the decision boundary separating class  $w_i$  from  $w_j$ . Thus  $d_{ij} > 0$  for pattern of class  $w_i$  and  $d_{ij} < 0$  for patterns of class  $w_j$ .

$$d_j(\mathbf{x}) = \mathbf{x}^T \mathbf{m}_j - \frac{1}{2} \mathbf{m}_j^T \mathbf{m}_j, \quad j = 1, 2, \dots, M \quad (11)$$

$$\mathbf{m}_j = \frac{1}{N_j} \sum_{\mathbf{x} \in w_j} \mathbf{x}, \quad j = 1, 2, \dots, M \quad (12)$$

where  $M$  is the number of pattern classes,  $N_j$  is the number of pattern vectors from class  $w_j$  and the summation is taken over these vectors.

Support Vector Machines (SVMs) have recently been known to be successful in a wide variety of applications [10][15]. SVM-based and WED-based classifier are also compared in this work. In SVM, we first have a training data set, like,  $D = \{(x_i, y_i) | x_i \in X, y_i \in Y, i = 1, \dots, m\}$ . Where  $X$  is a vector space of dimension  $d$  and  $Y = \{+1, -1\}$ . The basic idea of SVM consists in first mapping  $x$  into a high dimension space via a function, then maximizing the margin around the separating hyperplane between two classes, which can be formulated as the following convex quadratic programming problem:

$$\text{maximize} \quad W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (K(x_i, x_j) + \frac{1}{C} \delta_{i,j}) \quad (13)$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, \forall i, \quad (14)$$

$$\text{and} \quad \sum_i^m y_i \alpha_i = 0 \quad (15)$$

where  $\alpha_i (\geq 0)$  are Lagrange multipliers.  $C$  is a parameter that assigns penalty cost to misclassification of samples.  $\delta_{i,j}$  is the Kronecker symbol and  $K(x_i, x_j) = \langle \phi(x_i) \cdot \phi(x_j) \rangle$  is the Gram matrix of the training examples. The form of decision function can be described as

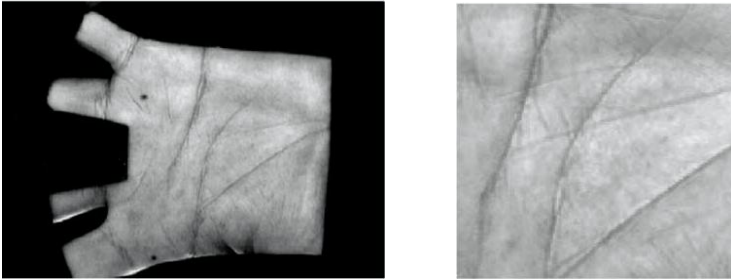
$$f(x) = \langle w, \Phi(x) \rangle + b \quad (16)$$

where,  $w = \sum_{i=1}^m \alpha_i^* y_i \Phi(x_i)$ , and  $b$  is a bias term.

## 5 Experiments

The PolyU palmprint database [9] was obtained by collecting palmprint images from 193 individuals using a palmprint capture device. People was asked to provide about 10 images, each of the left and right palm. Therefore, each person provided around 40 images, so that this PolyU database contained a total of 7,752 grayscale images from 386 different palms. The samples were collected in two sessions, where the first ten samples were captured in the first session and other ten in the second session. The average interval between the first and second collection was 69 days. The resolution of all original palmprint images is 384 x 284 pixels at 75 dpi. In addition, they changed the light source and adjusted the focus of the CCD camera so that the images collected on the first and second occasions could be regarded as being captured by two different palmprint devices. The palmprint images collected in the second occasion were also captured under different lighting conditions.

At the experiments on the database, we use the preprocessing technique described in [9] to align the palmprints. In this technique, the tangent of the two holes (they are between the forefinger and the middle finger, and between the ring finger and the little finger) are computed and used to align the palmprint. The central part of the image, which is 128 x 128, is then cropped to represent the whole palmprint. Such preprocessing greatly reduces the translation and



**Fig. 3.** Original palmprint and it's cropped image

rotation of the palmprints captured from the same palms. An example of the palmprint and its cropped image is shown in Figure 3.

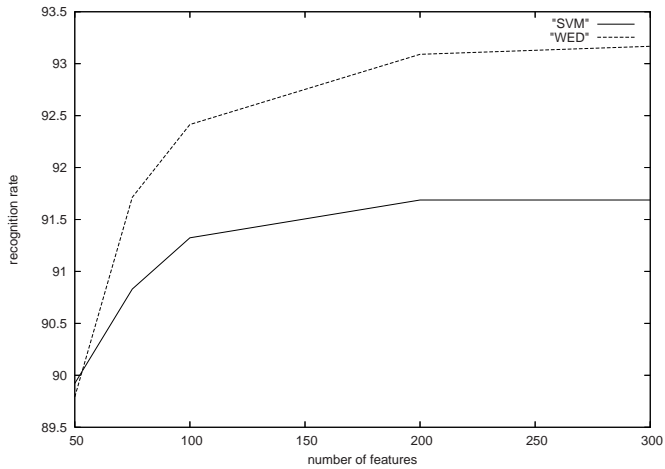
In the first experiment on the database, the first session was used as training set, second session includes 3850 samples of 386 different palms was also used as testing set. In this experiment, the features are extracted by using the proposed kernel based eigenspace method with length 50, 75, 100, 200, and 300. Weighted Euclidean distance(WED)-based matching was used to cluster those features. The matching is separately conducted and the results are listed in Table 1. The numbers given in Table 1 correspond to the correct recognition samples in all test samples (3850). The entries in brackets also represent the corresponding recognition rate. High recognition rate **93.168%** was achieved for the DWT+KPCA with feature length of 300. A nearest-neighbor classifier based on WED is employed to produce recognition rates given in the Table 1. The recognition rates obtained by PCA and kernel PCA based methods are comparatively illustrated in Table 1. When the feature number varies from 50 to 300, although KPCA-based approach only achieves higher recognition rate than PCA-based with feature length of 75, but DWT+KPCA based the proposed method achieved higher recognition rate then all combinations of PCA-based and FFT+KPCA-based approaches. Finally, it is evident that feature length can play an important role in the matching process. Long feature lengths lead to a high recognition rate.

**Table 1.** Comparative performance evaluation for the different matching schemes with different feature lengths. Train is first session, test is second session.

Method	Feature length				
	50	75	100	200	300
PCA	3411 (88.597)	3477 (90.311)	3498 (90.857)	3513 (91.246)	3513 (91.246)
DWT+PCA	3444 (89.454)	3513 (91.246)	3546 (92.103)	3570 (92.727)	3568 (92.675)
KPCA	3411 (88.597)	3481 (90.415)	3498 (90.857)	3508 (91.116)	3510 (91.168)
FFT+KPCA	2746 (71.324)	2933 (76.181)	3034 (78.805)	3174 (82.441)	3253 (84.493)
DWT+KPCA	3457 (89.792)	3531 (91.714)	3558 (92.415)	3584 (93.09)	3587 (93.168)

The performance variation for WED-based nearest-neighbor (NN) and SVM classifiers with the increase in number of features are shown in Figure 4. The SVM using radial basis function was employed in the experiments and the parameters of SVM were empirically selected. The training parameter  $\gamma$ ,  $\epsilon$  and  $C$  were empirically fixed at 0.55, 0.001, and 100, respectively. As shown in Figure 4, the SVM classifier achieved higher recognition when 50 features were only implemented. For the feature lengths longer than 50, the WED-based NN classifier has achieved better performance.

As final experiment and very similar to the experiments published in literature, the palm images collected from the first session were only used to test the proposed algorithm. We use the first four palmprint images of each person as training samples and the remaining six palmprint images as the test samples. So, the numbers of training and test samples are 1544 and 2316. We also test the

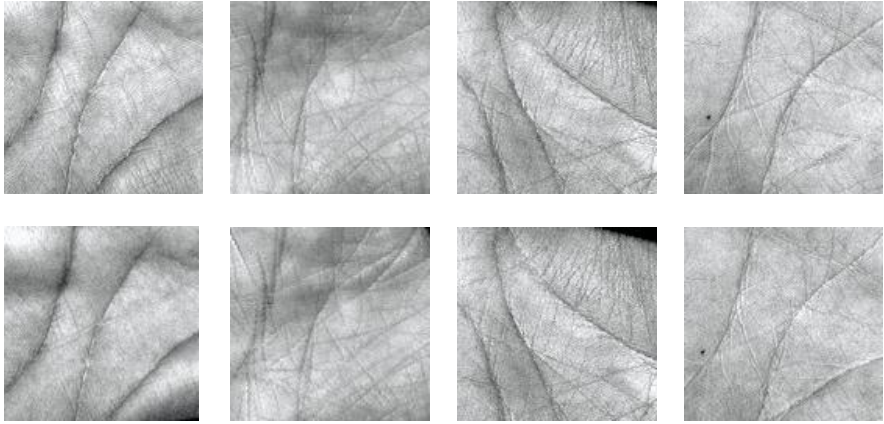


**Fig. 4.** Performance analysis of classifier with the number of features: DWT+ KPCA method using the SVM- and WED-based classifiers

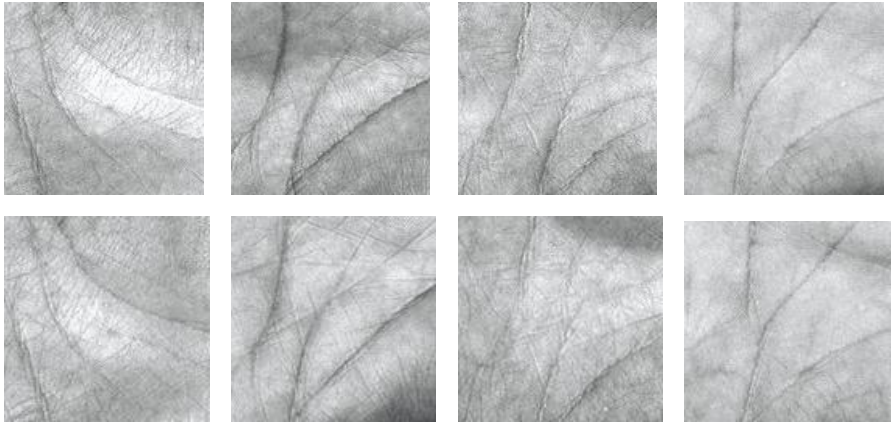
**Table 2.** Testing results of the eight matching schemes with different feature lengths

Method		Feature length				
		50	100	200	300	380
PCA	LED	60.664 %	71.804 %	74.568 %	1723 (74.395 %)	1717 (74.136 %)
	WED	98.747 %	99.179 %	99.093 %	2294 (99.05 %)	2292 (98.963 %)
DWT + PCA	LED	59.542 %	71.459 %	87.305 %	2032 (87.737 %)	2032 (87.737 %)
	WED	98.834 %	99.309 %	99.352 %	2301 (99.352 %)	2302 (99.395 %)
KPCA	LED	63.557 %	73.661 %	75.82 %	1730 (74.697 %)	1712 (73.92 %)
	WED	98.877 %	99.222 %	99.05 %	2293 (99.006 %)	2291 (98.92 %)
DWT KPCA	LED	83.462 %	86.01 %	86.01 %	2025 (87.435 %)	2039 (88.039 %)
	WED	98.747 %	99.309 %	99.568 %	2308 ( <b>99.654 %</b> )	2308 ( <b>99.654 %</b> )

8 approaches against conventional PCA method using different test strategies. Based on these schemes, the matching is separately conducted and the results are listed in Table 2. The meaning of LED and WED in Table 2 is linear Euclidean discriminant and weighted Euclidean distance based nearest neighbor classifier, respectively. The numbers given for feature lengths 300 and 380 in Table 2 represent the number of the correct recognition samples in all 2316 palms used as test samples. The entries in the brackets also represent the corresponding recognition rate (%). A high recognition rate (**99.654 %**) was achieved for kernel PCA with 2D-DWT (abbreviated as DWT+KPCA) and WED classifier approach, with feature length of 300. One of the important conclusion from Table 2 is that, long feature lengths lead to a high recognition rate. However, this principle only holds to a certain point, as the experimental results summarized in Table 2 show that the recognition rate remain unchanged, or even become worse, when the feature length is extended further.



**Fig. 5.** Experimental results by the different rotation and translation conditions. (Top) Some palm images in training set, (Bottom) Correctly classified corresponding samples in testing set.



**Fig. 6.** Misclassified four palm samples. Top: Some palm images in training set, Bottom: Corresponding and misclassified samples in testing set.

Typical samples in this database are shown in Figs. 5 in which the top images were used as training samples, the bottom images were also used as test samples. Although the rotation and translation conditions are quite different from the samples used as test set, the proposed algorithm can still easily recognize the same palm. The misclassified samples were only 8 samples in all 2316 used as testing set, and some of them are also shown in Figure 6 in which the top images show the sample in training set, and corresponding bottom images were misclassified samples used in test set. The other misclassified four samples have not been shown because of the page limitation.

**Table 3.** Comparison of different palmprint recognition methods

		Method									
		Proposed	In [4]	In [5]	In [3]	In [8]	In [6]	In [7]	In [16]	In [17]	In [18]
Database	palms	386	3	100	300	382	160	100	100	190	50
	samples	3860	30	200	3000	3056	1600	600	1000	3040	200
Recog.	Rate	<b>99.654</b>	95	91	99.2	99.149	97.25	97.5	95.8	98.13	98

Comparison has been finally conducted among our method and other methods published in literature, and is illustrated in Table 3. The databases given in the Table 3 are defined as the numbers of the different palms and whole samples tested. The data represent the recognition rates given in Table 3 is taken from experimental results in the cited papers. In biometric systems, the recognition accuracy will decrease dramatically when the number of image classes increase [1]. Although the proposed method is tested on the public database includes more different palms and samples, the recognition rate of our method is more efficient, as illustrated in Table 3.

## 6 Conclusion

This paper presents a new appearance-based non-linear feature extraction (kernel PCA) approach to palmprint identification that uses low-resolution images. We first transform the palmprints into wavelet domain to decompose the original palm images. The kernel PCA method is then used to project the palmprint image from the very high-dimensional space to a significantly lower-dimensional feature space, in which the palmprints from the different palms can be discriminated much more efficiently. WED based NN classifier is finally used for matching. The feasibility of the wavelet-based kernel PCA method has been successfully tested on PolyU database. The data set consists of 7752 images of 386 subjects. Experimental results show the effectiveness of the proposed algorithm for palmprint recognition.

**Acknowledgments.** This research is partially supported by The Research Foundation of Karadeniz Technical University (Grant No: KTU-2004.112.009.001). The authors would like to thank to Dr. David Zhang from the Hong Kong Polytechnic University, Hung Hom, Hong Kong, for providing us with the PolyU palmprint database.

## References

1. Zhan D., Jing X., Yang J.: Biometric Image Discrimination Technologies. Computational Intelligence and Its Application Series, Idea Group Publishing, (2006) 80–95
2. Zhang D., Shu W.: Two Novel Characteristics in Palmprint Verification: Datum Point Invariance and Line Feature Matching. Pattern Recognition, Vol. 32, (1999) 691–702

3. Wu X., Zhang D., Wang K.: Fisherpalms Based Palmprint Recognition. *Pattern Recognition Letters*, Vol. 24, Issue 15, November, (2003) 2829–2838.
4. Duta N., Jain A.K., Mardia K.V.: Matching of Palmprint. *Pattern Recognition Letters*, Vol. 23, No.4, (2002) 477–485
5. You J., Li W., Zhang D.: Hierarchical Palmprint Identification via Multiple Feature Extraction. *Pattern Recognition*, Vol. 35 No.4, (2002) 847–859
6. Wang Y., Ruan Q., Kernel Fisher Discriminant Analysis for Palmprint Recognition. *IEEE The 18th Int. Conf. on Pattern Recognition, ICPR'06*, (2006) 457–460
7. Jiang W., Tao J., Wang L.: A Novel Palmprint Recognition Algorithm Based on PCA and FLD. *IEEE, Int. Conf. on Digital Telecommunications*, August, (2006) 28–32
8. Lu G., Zhang D., Wang K.: Palmprint Recognition Using Eigenpalms Features. *Pattern Recognition Letters*, Vol. 24. Issue 9-10, June (2003) 1463–1467
9. Zhang D., Kongi W., You J., Wong M.: Online Palmprint Identification. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 25, No.9, September (2003) 1041–1049
10. Scholkopf B., Somala A.: *Learning with Kernel: Support Vector Machine, Regularization, Optimization and Beyond*. MIT Press, (2002).
11. Daubechies I.: *Ten Lectures on Wavelets*. Philadelphia., PA: SIAM, (1992).
12. Chang T., Kuo C.J.: Texture Analysis and Classification with Tree-Structured Wavelet Transform. *IEEE Trans. Image Processing*, Vol.2, No.4, (1993) 429–441
13. Zhang B., Zhang H., Sam S.: Face Recognition by Applying Wavelet Subband Representation and Kernel Associative Memory. *IEEE Transactions on Neural Networks*, Vol. 15, No. 1, (2004) 166–177
14. Chien J., Wu C., Discriminant Waveletfaces and Nearest Feature Classifier for Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 12, (2002) 1644–1649.
15. Li W., Gong W., Yang L., Chen W., Gu X.: Facial Feature Selection Based on SVMs by Regularized Risk Minimization. *IEEE, The 18th Conference on Pattern Recognition (ICPR'06)*, (2006) 540–543
16. Kumar A., Zhang D.: Personal Recognition Using Hand Shape and Texture. *IEEE Transactions on Image Processing*, Vol. 5, No. 8, (2006) 2454–2460
17. Jing X.Y., Zhang D.: A Face and Palmprint Recognition Approach Based on Discriminant DCT Feature Extraction. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 34, No. 6 (2004) 2405–2415
18. Zhang L., Zhang D.: Characterization of Palmprints by Wavelet Signatures via Directional Context Modeling. *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 34, No. 3, (2004) 1335–1347
19. Valentin T.: *Face-space Models of Face Recognition*. In *Computational, Geometric, and Process Perspectives on Facial Cognition: Context and Challenges*, Hillsdale, NJ: Lawrence Erlbaum, (1999)



# ICS: An Interactive Classification System

Yan Zhao, Yiyu Yao, and Mingwu Yan

Department of Computer Science, University of Regina  
Regina, Saskatchewan, Canada S4S 0A2  
{yanzhao,yyao}@cs.uregina.ca, mervin.yan@gmail.com

**Abstract.** Interactive data mining focuses on efficient and effective human-computer interactions for data analysis purposes. An interactive system is an integration of a human user and a computer machine. ICS, an interactive classification system, is implemented to demonstrate the power of interactive data mining. The interaction is mutually beneficial to users and machines. This article describes the architecture of ICS, and introduces the main features of ICS in the entire data mining process.

## 1 Introduction

Data mining deals with many important tasks, such as description, prediction and explanation of data. Whereas many data mining models concentrate on automation and efficiency, interactive data mining systems focus adaptive and effective communications between human users and computer systems. One of the key elements that distinguishes an interactive data mining systems from more traditional data mining systems is its capability to dynamically support a user's expectation, mining and reasoning. Though human-machine interaction has been emphasized for many disciplines, it did not get enough attention in the domain of data mining until recently [1,2,4,8]. The research includes several related areas: the input of domain knowledge, the input of user requirements and the enhancement of visualization.

The knowledge discovery process is normally described as six phases: data preparation, data selection and reduction, data pre-processing and transforming, pattern discovery, pattern explanation and evaluation, and pattern presentation [2,3,5,6]. Users need to involve in all these phases. Specifically, users expect computer systems to support different forms of user involvement, such as: hypothesis formulation, information acquisition, assistance acquisition, manipulation, and evaluation and explanation [7]. These forms of involvement can be applied to the entire data mining process to arrive at desirable mining results. Through interaction and communication, computers and users can share the tasks involved in order to achieve a good balance of automation and human control. Moreover, interactive data mining can encourage users' learning, improve insight and understanding of the problem to be solved, and stimulate users to explore creative possibilities. Users' feedback can be used to improve the system. The interaction is mutually beneficial, and imposes new coordination demands on both sides [7,8].

To exemplify the power of interactive data mining, ICS, an interactive classification system, is designed and implemented (refer to the current version ICS-V1 on <http://www.cs.uregina.ca/~yanzhao/ICS.html>). The main output of ICS is a set of classification rules represented as a spacial tree structure, called a granule tree. ICS enables users to be involved in two main issues of classification. First, it allows a user to visually select and state *which granule to be classified*. The user can express his/her own interest or priority to execute the classification by a free selection of the candidate granule to be classified. Second, the system allows the user to visually select and state *how to classify* by providing diverse statistical evaluations. Various interactions are supported in all of the six data mining phases.

## 2 An Architecture of Interactive Data Mining Systems

In this section, we discuss six phases of interactive data mining, and five basic forms of human-user interactions. Based on this model, we outline the architecture and major components of ICS.

### 2.1 The Process of Interactive Data Mining

In an interactive system, the six phases of data mining can be carried out as follows:

- *Interactive data preparation* prepares dataset to be processed, and observes raw data within a specific format.
- *Interactive data selection and reduction* involves the reduction of the number of attributes and/or the number of records. A user can specify the attributes of interest and/or data area, and remove data that is outside of the area of concern.
- *Interactive data pre-processing and transformation* determines the number of intervals, as well as cut-points for continuous datasets, and transforms the dataset into a workable dataset.
- *Interactive pattern discovery* interactively discovers patterns under the user's guidance, selection, monitoring and supervision. Interactive controls include choosing search strategies and heuristics, deciding rule directions, and handling abnormal situations.
- *Interactive pattern explanation and evaluation* explains and evaluates the discovered pattern if the user requires it. The effectiveness and usefulness of the explanations is subject to the user's judgement.
- *Interactive pattern presentation* visualizes the patterns that are perceived during the pattern discovery phase, and/or the pattern explanation and evaluation phase.

Practice has shown that the process is virtually a loop, which is iterated until satisfying results are obtained.

## 2.2 The Forms of Interactive Data Mining

Users should be allowed to formulate hypotheses, describe decisions and selections based on their preference and judgement. For example, a user can state an interested class for classification tasks, express a target knowledge representation, indicate a question, infer features for explanation, describe a preference order of attributes, set up constraints, and so on. Subjects of hypotheses differ among the varying views of individuals. The potential value consideration enters into the choice of proposition.

Information can be presented in various fashions and structures in an interactive data mining system. Raw data is raw information. Mined rules are extracted knowledge. Numerous measurements show the information of an object from different aspects. Each data mining phase contains and generates much information. An object might be changed; the information it holds might be erased, updated or manipulated by the user in question. Benchmarks, standards and de facto standards are valuable reference knowledge, which can make it easier to learn and evaluate new applications. Users need to retrieve information in an interactive manner.

One of the roles that an interactive system can play is to provide knowledge or skills that the user does not have in-house, for example, doing an evaluation or providing an analysis of the implications of environmental trends. To achieve this expert role, the interactive system must be able to “understand” the human proposition, and be able to make corresponding inferences. Assistance is especially useful while the domain is complex and the search space is huge. It ensures the process develops in a more balanced manner.

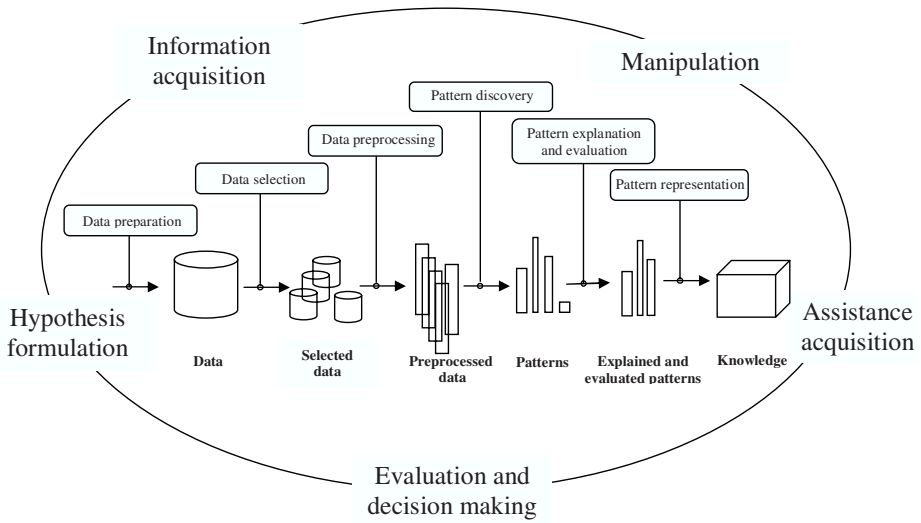
Manipulation is the form of user participation that includes selecting, retrieving, combining and changing objects, and using operated objects to obtain new results. Different data mining phases require different kinds of manipulations. Interactive systems should allow users to build their own models, and define their own heuristics and algorithms. The procedures can be connected by functions similar to the pipe command in UNIX systems. It means that the standard output of the left of the pipe is sent as standard input of the right of the pipe.

The evaluation and explanation phase is guided by humans. Sometimes people doubt and challenge the so-called knowledge, if it is not understandable within their previous knowledge. We believe that information can turn into knowledge if, and only if, it can be rationalized, explained and validated.

A particular interactive data mining system can involve interactions of all the five forms at the six different phases. Figure 1 illustrates the process and the forms of interactive data mining.

## 2.3 The Architecture of ICS

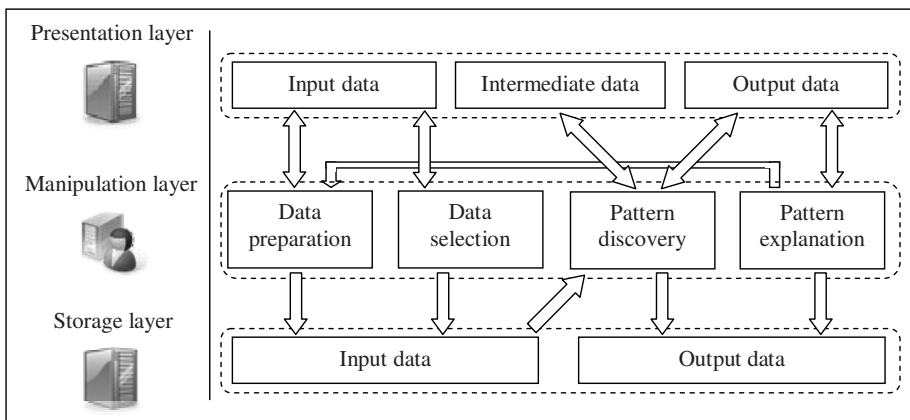
Based on the framework of interactive data mining, an interactive classification system, ICS, is developed. The general architecture of ICS is shown in Figure 2. It is composed of three layers: the storage layer, the interactive manipulation layer and the presentation layer. The bottom layer is executed by the computer, storing raw user data and classification results. The middle layer, implementing



**Fig. 1.** Process and form-based interactive data mining

the main phases of data mining, is executed by the user and the computer. The upper layer is supported by the computer. It presents raw data, intermediate results, and final results in different forms.

The manipulation layer is the most essential part of ICS. Through the data preparation component, users can define the schema of a new table, create tables, import tables and edit data in the existing tables. These tables are stored in the storage layer, and presented in the presentation layer. Through the data selection component, users can split the table into two parts. One is for



**Fig. 2.** The architecture of ICS

training a classifier, the other is for testing the classifier. The partitioned tables are also stored and presented. Through the pattern discovery component, users can select decision classes of interest, select classification and evaluation measures, select algorithms, determine searching strategies and heuristics, decide stopping criteria, and evaluate classification rules at any time. The intermediate results are presented in different forms, such as a tree, a table or a chart. These results help the user with further decision-making. If the user questions the reason of a particular discovered result, he/she can construct explanations of it through the pattern explanation component. This component allows the user to suggest explanation profiles, and construct an explanation table. The plausible explanations can be constructed in the explanation table through the pattern discovery component. We introduce the detail modules of the architecture in the next section.

### 3 Implementation of ICS

We use an information table as our data representation, and a granule tree as our knowledge representation. After introducing the basic concepts of information tables and granule trees, we present the main features of ICS based on the system architecture.

#### 3.1 Data and Knowledge Representation

An information table is the following tuple:

$$S = (U, At, \{V_a \mid a \in At\}, \{I_a \mid a \in At\}, L),$$

where  $U$  is a finite nonempty set of objects,  $At$  is a finite nonempty set of attributes,  $V_a$  is a nonempty set of values of  $a \in At$ ,  $I_a : U \rightarrow V_a$  is an information function.  $L$  is a decision logic language, consists of a set of formulas. An atomic formula of  $L$  is a descriptor  $a = v$ , where  $a \in At$  and  $v \in V_a$ . The well-formed formulas (wff) of  $L$  is the smallest set of formulas containing the atomic formulas and closed under  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\rightarrow$  and  $\equiv$ . If a formula is defined by only logical conjunction  $\wedge$ , it is called a *conjunctive*. If a formula is defined by only logical disjunction  $\vee$ , it is called a *disjunctive*.

Given a formula  $\phi \in L$ , the set  $m(\phi)$ , defined by  $m_S(\phi) = \{x \in U \mid x \models \phi\}$ , is called the meaning of the formula  $\phi$  in  $S$ , or a definable granule of  $S$ . If  $\phi$  is a conjunctive, then  $m_S(\phi)$  is called a conjunctively definable granule. If  $\phi$  is a disjunctive, then  $m_S(\phi)$  is called a disjunctively definable granule.

For classification tasks,  $At = C \cup \{class\}$ , where  $C$  is a set of conditional attributes and  $class$  is a decision attribute. A classification rule is expressed as  $\phi \Rightarrow \psi$ , where the formula  $\phi$  is defined by conditional attributes in  $C$  and the formula  $\psi$  is defined by the decision attribute  $class$ .

A *granule network* [8] systematically organizes all the conjunctively definable granules and conjunctives with respect to the given universe. Each node consists of a granule, and each arc leading from a granule to its child is labelled by an atomic

formula. A path from a coarse granule to a fine granule indicates a conjunctive relation. Given an information table  $S = (U, At, \{V_a\}, \{I_a\})$ , a granule network has  $|At| + 1$  levels. According to a hierarchical structure, the root node is the most coarse granule  $U$ , labelled by  $\top$ . The second level contains all the 1-conjunctive definable granules, the third level contains all the 2-conjunctive definable granules, and so on, until the  $(|At| + 1)^{th}$  level contains all the  $|At|$ -conjunctive definable granules.

A *granule tree*, as a heuristic searching result for classification, is a portion of a granule network. It can be organized as a tree structure. Each path from the root to a leaf can be equivalently expressed as a decision rule.

### 3.2 Interactive Data Preparation and Selection

Data can be stored in ICS in three ways. First, ICS allows a user to define and create his/her datasets from scratch. A “schema editor” module allows the user to specify the attribute names of a dataset. A “data editor” module allows the user to type in each individual data entry, or copy-paste an existing dataset from elsewhere. Second, ICS can import an existing dataset from Microsoft Access or Excel directly, and convert it to XML format. All the user datasets in ICS can be edited before being processed in the “data editor” module. Third, a user can compose a new dataset by natural joining two or more than two existing tables together in the “query builder” module. The process of selecting tables to be joined is subjectively based on users’ requirement.

The natural join operation designed in the “query builder” module is especially useful for explaining data mining results. The explanation profiles of a target to be explained may exist inside or outside of the original dataset. By natural joining the tables in which a pattern is located, and another table in which the proposed explanation profiles are stored, we are able to construct plausible explanations of the discovered pattern. Intuitively, given different explanation profiles, different sets of explanatory accounts can be generated.

For the purpose of training and testing a classifier, a user can decide either to use a random leave-out method, or the 5-cross validation method to divide the dataset into two parts. For the random leave-out method, ICS randomly picks objects for constructing classification rules or trees, and keeps the remaining objects for testing the accuracy of the learned rules or the tree. For the 5-cross validation method, ICS needs the user to indicate which fold is retained for testing, and the remaining four folds are used for training.

### 3.3 Interactive Pattern Discovery

The discovery module is called a “data analyser,” which consists of three sub-modules: low order rule miner, high order rule miner and built-in algorithm rule miner. Each sub-module can generate a specific granule tree, and can be evaluated separately. Either a partition space or a covering space of the granule network can be searched for classification solutions. Many measures are involved in the classification process. They represent different aspects of all formulas that can be added conjunctively to the selected granule. ICS assists the user

**Input:** A dataset for classification.

**Output:** A granule tree and its corresponding classification rules.

Set  $U$  as the root node of a granule tree at the initial stage.

While *the user wants to continue*

    Select a branch  $N$ .

    If ADD

        Select an atomic formula  $a = v$  with respect to  $N$  regarding a certain criterion.

        Modify the granule tree by adding the granule  $m(\phi \wedge a = v)$  as a new node, connect it to  $N$  by an arc, and name it by  $a = v$ .

        Label the node by the class that satisfies the majority objects of the node.

    If DELETE

        Modify the granule tree by deleting all the finer granules of  $N$ , removing the arcs, and deleting the granule  $N$  from the tree.

    Update information of the branch  $N$  and the constructed tree.

**Fig. 3.** An algorithm of interactive classification

to observe all these aspects at the same time, and allows the user to select the measures that are preferred. The induction process of ICS is briefly outlined in Figure 3.

**Information acquisition.** In the process of interactive pattern discovery, users require three types of information before deciding what action to take.

- Information of the constructed granule tree. At the initial stage, the granule tree only has the root, the entire universe, the biggest granule. If it is not consistently classified by any class, we start the classification process. Nodes from the root to each labelled leaf form a classification rule. The view of the granule tree and the view of the corresponding set of rules are switchable. The evaluation of the rule set is also the evaluation of the tree.
- Information of a selected branch. A branch is a path from the root to a selected node. Initially, the only branch is the root. After an atomic formula is added conjunctively to the root, there are two branches, the root, and the branch ended with the new added node. The user can choose any one of the branches to investigate. As a result, he/she can alternatively add another atomic formula to the previous one, or add an atomic formula to the root. Some measures can be selected to evaluate the properties of the selected branch.
- Information of the available atomic formulas (called active nodes in ICS) and the available attributes (called active attributes in ICS) with respect to a selected branch. When a branch is consistently classified, the finer granules of it are all consistently classified, and thus are not meaningful. When a branch is not yet consistently classified, then the user needs to decide which atomic formula should be added conjunctively to the selected branch.

Measures of active attributes:

$$\text{NumberOfAttributeValues}(a) = |V_a|;$$

$$\text{ConditionalEntropy}(\text{class}|a)$$

$$= - \sum_{v \in V_a} p(a = v) \sum_{c_i \in V_{\text{class}}} p(\text{class} = c_i | a = v) \log(\text{class} = c_i | a = v),$$

$$= - \sum_{v \in V_a} \sum_{c_i \in V_{\text{class}}} p(\text{class} = c_i, a = v) \log(\text{class} = c_i | a = v);$$

$$\text{JointEntropy}(\text{class}, a)$$

$$= - \sum_{v \in V_a} \sum_{c_i \in V_{\text{class}}} p(\text{class} = c_i, a = v) \log(\text{class} = c_i, a = v);$$

$$\text{Gini}(a)$$

$$= 1 - \sum_{v \in V_a} p(a = v) \sum_{c_i \in V_{\text{class}}} p^2(\text{class} = c_i | a = v).$$

Measures of active nodes:

$$\text{confidence}(a = v \Rightarrow \text{class} = c_i) = p(\text{class} = c_i | a = v);$$

$$\text{coverage}(a = v \Rightarrow \text{class} = c_i) = p(a = v | \text{class} = c_i);$$

$$\text{Entropy}(a = v) = - \sum_{c_i \in V_{\text{class}}} p(\text{class} = c_i | a = v) \log(\text{class} = c_i | a = v);$$

$$\text{Generality}(a = v) = |m(a = v)|.$$

**Mining low order and high order rules.** Two levels of dependencies, referred to as the *local* and *global* dependencies, may be observed in an information table. The local dependencies show how *one* specific combination of values on one set of attributes determine *one* specific combination of values on another set of attributes. The global dependencies show how *all* combinations of values on one set of attributes determine *all* combinations of values on another set of attributes. Given an information table  $S$ , for  $A, B \subseteq At$ , a low order rule and a high order rule are defined as:

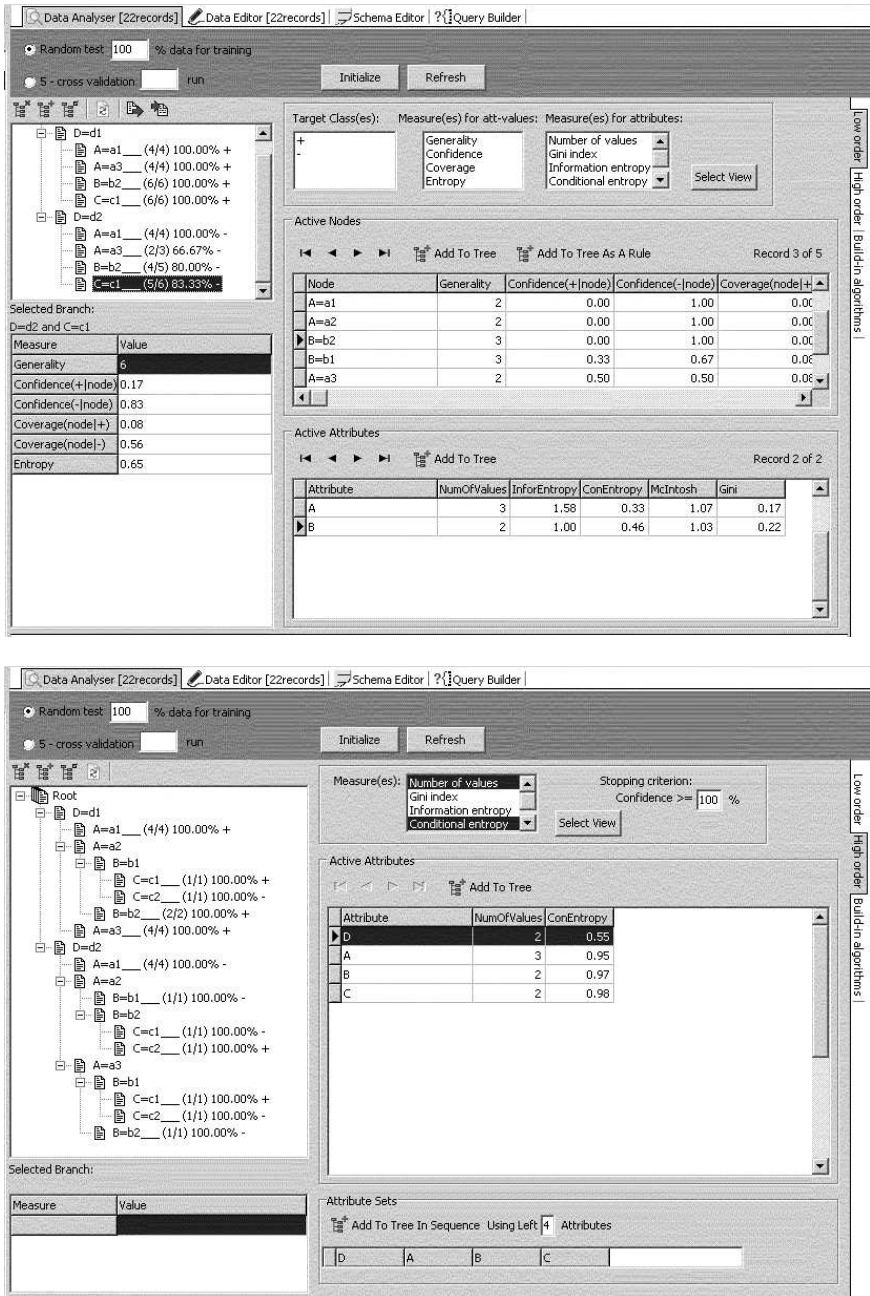
$$\text{A low order rule : } \forall_{a \in A} I_a(x) = v_a \Rightarrow \forall_{b \in B} I_b(x) = v_b.$$

$$\text{A high order rule : } \forall_{a \in A} I_a(x) = I_a(y) \Rightarrow \forall_{b \in B} I_b(x) = I_b(y).$$

The interface of the low order rule module and the high order rule module is illustrated in Figure 4, together with two constructed granule trees. ICS supports two different ways of mining low order classification rules. With respect to a selected tree node, one can either add an active node, or add an active attribute. For the former, only one rule is generated at each time; while for the latter, there is a set of rules, corresponding to the possible values of the added attribute, being added simultaneously. Both ways only concern the selected node, and thus are low ordered.

ICS also supports two ways for mining high order classification rules. One way is to select an active attribute at each time, and add it to all the inconsistent nodes of the tree at once. Each node derives a set of rules, corresponding to the possible values of the added attribute. The other way is to decide an ordering of the attribute set. Following this order, ICS adds the attributes one by one to the inconsistent nodes of the tree. By clicking one title of the active attribute table, the attributes are sorted according to the selected measure, and the derived attribute order is shown in the attribute order table on the right bottom. Both





**Fig. 4.** ICS produces a low order granule tree and a high order granule tree in the data analyser module, where 100% of user data are used for training. Both the decision classes and all the measures are selected to be shown for the low order module, while only two measures are selected for the high order module.

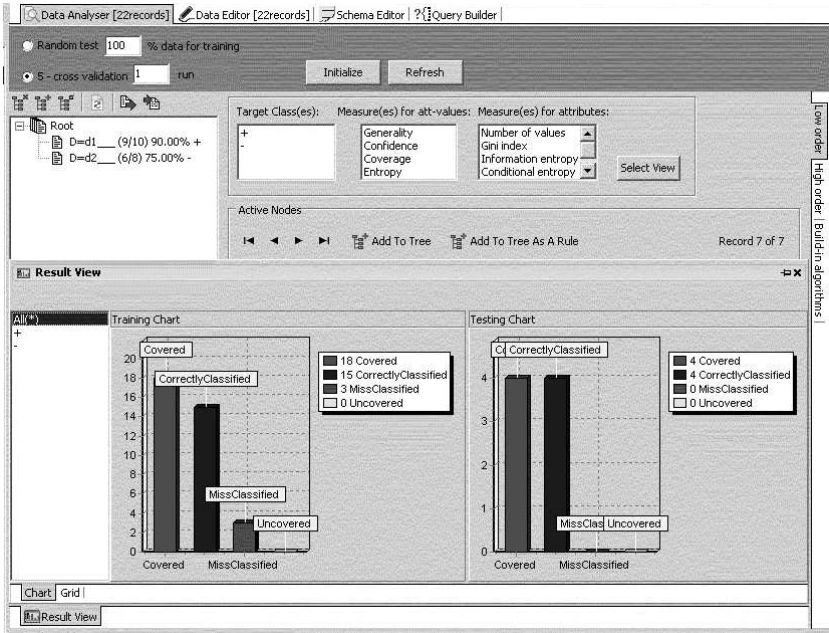
ways concern all the inconsistent nodes at the same time. They thus are high ordered. A pre-pruning method is provided by ICS to allow the user to indicate the confidence threshold. Although a high order tree is constructed more quickly, there is less human interaction involved in the discovery process.

**Mining partition-based and covering-based granule trees.** Partitions and coverings are two simple and commonly used granulations of the universe. A partition of a finite universe  $U$  is a collection of non-empty, and pairwise disjoint granules of  $U$  whose union is  $U$ . A covering of a finite universe  $U$  is a collection of non-empty, possibly overlapped granules of  $U$ , whose union is  $U$ . Partitions are a special case of coverings.

To describe and classify all the classes, partition-based granule trees are suitable. Whenever an attribute is selected, all of its possible values are added to the granule tree, with the corresponding granules being pairwise disjointed. The partition-based approach ensures that no portion will be missed for classification. If one needs to achieve partial success, namely, to describe and classify one particular class, then covering-based granule trees are more suitable. In this case, the active nodes and their measurements are of concern. Normally, the progression of investigation moves from the most promising branch to the less promising ones. One can apply the depth-first mode to explore each branch sequentially, or apply the breath-first mode to explore the granules at the same level. A mixture method also is allowed.

**Mining rules and exceptions.** A classification rule is expressed as  $\phi \Rightarrow \psi$ . When the accuracy of a rule is not satisfied, there are two methods to obtain a rule with higher confidence. The first method is to construct a rule in the form of  $\phi \wedge \phi' \Rightarrow \psi$ . That means, by refining the granule of  $m(\phi)$  to the sub-granule  $m(\phi \wedge \phi')$ , one expects the finer granules are tend to be more consistent. The second method is to construct a rule in forms of  $\phi \wedge \phi' \Rightarrow \psi'$ . The interpretation is: if only the condition  $\phi$  is known, we can conclude  $\psi$  to a certain degree. However, if we also know that  $\phi'$  holds, we need to change our conclusion to  $\psi'$ . That is, the second rule is an exception rule to the first general rule. By using an interactive approach, ICS can generate both conventional classification rules and exception rules.

**Interactive system assistances.** ICS provides effective assistances for interactive classification. Besides the functions we have introduced above, there is an important sub-module of data analyser, called the “built-in algorithms” module. It contains various prototypes of partition-based algorithms, such as ID3, ASSISTANT, C4.5, CART, and covering-based algorithms like PRISM, CN2 and Itrule. Users can compare the resulting trees with these standard algorithms, and improve and adjust the current learning results. The built-in algorithm module is, of course, faster than the low order rule module and the high order rule module. However, it supports the lowest level of human-computer interactivity.



**Fig. 5.** ICS evaluates a constructed low order tree and its corresponding rules in the data analyser module, low order rule mining sub-module. 5-cross validation method is applied for splitting user data. The first fold of data records are for testing. Both the decision classes, and both the training and testing tables are examined.

### 3.4 Interactive Pattern Evaluation

In the process of interactive evaluation, users need to obtain two types of evaluations: on one hand, we need to evaluate the description of the training data. ICS keeps track of the number of objects that have been processed and covered by any one of the constructed rules, the number of objects that have not been covered by any rules, the sum of the correctly classified objects that covered by certain solutions and/or manually-set solutions, and the sum of incorrectly classified objects. The values are updated when a new node is added to a selected branch, a label is manually set for a branch, or a branch is deleted from the granule tree. On the other hand, we need to evaluate the prediction of the testing data regarding the constructed classification rules. By the interactive method, one does not need to complete the whole training process before doing the test. The test process can be carried out whenever the user wishes. Similarly, the classification process can be stopped manually when the accuracy of the testing result is acceptable.

Corresponding to the three sub-modules: low order, high order and built-in algorithms, three separate evaluation modules, called “result view” are implemented. The user graphic interface of result view is illustrated in Figure 5,

showing both the evaluation of the training data and the testing data regarding a constructed low order rule tree and its corresponding rules.

## 4 Conclusion

The objective of interactive data mining is to provide an effective and efficiency human-user interaction for data analysis purposes. ICS, equipped with many features, supports interactive classification. It can simulate different algorithms and generate different types of rules. Because the user decision-making process can be totally indeterministic and unpredictable, the behavior of ICS is too rich for a nice mathematical model.

We plan to extend ICS by adding more interactive features, such as an interactive data preprocessing module. This will enable ICS to handle continuous data, and allow users to cluster data according to their requirements. This module is necessarily supported by better data visualization approaches. We also plan to add a feature selection module into ICS. In this case, the relevant attributes are suggested to be used for classification, while the irrelevant attributes are suggested to be ignored.

## References

1. Anderst, M., Human involvement and interactivity of the next generations' data mining tools, *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Santa Barbara, CA, 2001.
2. Brachmann, R. and Anand, T., The process of knowledge discovery in databases: a human-centered approach, *Advances in Knowledge Discovery and Data Mining*, AAAI Press & MIT Press, Menlo Park, CA, 37-57, 1996.
3. Fayyad, U.M. and Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R. (Eds.) *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1996.
4. Han, J., Hu, X. and Cercone, N., A visualization model of interactive knowledge discovery systems and its implementations, *Information Visualization*, 2(2), 105-125, 2003.
5. Mannila, H., Methods and problems in data mining, *Proceedings of International Conference on Database Theory*, 41-55, 1997.
6. Yao, Y.Y., Zhao, Y. and Maguire, R.B., Explanation-oriented association mining using rough set theory, *Proceedings of Rough Sets, Fuzzy Sets and Granular Computing*, 165-172, 2003.
7. Zhao, Y., Chen, Y.H. and Yao, Y.Y., User-centered interactive data, *Proceedings of the Sixth IEEE International Conference on Cognitive Informatics*, 457-466, 2006.
8. Zhao, Y. and Yao, Y.Y., Interactive user-driven classification using a granule network, *Proceedings of the Fifth International Conference of Cognitive Informatics*, 250-259, 2005.

# Fast Most Similar Neighbor Classifier for Mixed Data

Selene Hernández-Rodríguez, J. Francisco Martínez-Trinidad,  
and J. Ariel Carrasco-Ochoa

Computer Science Department  
National Institute of Astrophysics, Optics and Electronics  
Luis Enrique Erro No. 1, Sta. María Tonantzintla, Puebla, CP: 72840, Mexico  
{selehdez, ariel, fmartine}@inaoep.mx

**Abstract.** The nearest neighbor (*NN*) classifier has been a widely used technique in pattern recognition because of its simplicity and good behavior. To decide the class of a new object, the *NN* classifier performs an exhaustive comparison between the object to classify and the training set  $T$ . However, when  $T$  is large, the exhaustive comparison is very expensive and sometimes becomes inapplicable. To avoid this problem, many fast *NN* algorithms have been developed for numerical object descriptions, most of them based on metric properties to avoid comparisons. However, in some sciences as Medicine, Geology, Sociology, etc., objects are usually described by numerical and non numerical attributes (mixed data). In this case, we can not assume the comparison function satisfies metric properties. Therefore, in this paper a fast most similar object classifier based on search methods suitable for mixed data is presented. Some experiments using standard databases and a comparison with other two fast *NN* methods are presented.

## 1 Introduction

The *NN* algorithm [1] has been a widely used technique for classification problems. Given a new object  $Q$ , the *NN* classifier consists in finding the nearest object  $O_{NN}$  of  $Q$  from a training set  $T$ , using a distance function, and assigning the class of  $O_{NN}$  to  $Q$ . However, performing an exhaustive comparison between objects becomes impractical for applications where the training set  $T$  is large. To avoid this problem, many fast *NN* algorithms, which were designed for numerical object descriptions and most of them based on metric properties, have been developed.

However, in some sciences as Medicine, Geology, Sociology, etc., objects are described by numerical and non numerical data (mixed data). In this case, we can not assume the comparison function satisfies metric properties and therefore, we can not use most of the methods proposed for numerical objects. So, if a metric is not available but a comparison function that evaluates the similarity between a pair of objects could be defined, the objective would be to find the most similar neighbor (*MSN*) and use it for classifying.

The *MSN* classifier is based on a training set  $T$  of  $N$  objects. Each object is described by  $d$  attributes, which can be numerical or non numerical. Given a new object  $Q$  to classify, the goal consists in finding the *MSN* according to a comparison function

and assigning to  $Q$  the class of its  $MSN$ . The exhaustive search of the  $MSN$ , as occurs with the  $NN$ , could be very expensive if  $T$  is large. Therefore, in this paper we introduce a fast  $MSN$  classification method based on a tree structure, which is not based on any metric property of the comparison function.

The paper is organized as follows: section 2 provides a brief review of tree based fast  $NN$  algorithms. In section 2.1 and 2.2 two well known methods are detailed. In section 2.3 the comparison functions used in this work are described. In section 3 our fast  $MSN$  classifier is introduced. In section 4 we report experimental results obtained using our methods and the methods described in section 2. Finally, in section 5 we present some conclusions and future work.

## 2 Related Work

To avoid the exhaustive search, one of the most common strategies is the branch and bound technique. In a preprocessing step, the objects in  $T$  are organized in a tree structure. In the classification step, the tree is traversed to find the nearest neighbor. The speed up is obtained while the exploration of some parts of the tree is avoided. One of the first fast  $NN$  classifier, that use the branch and bound technique, was proposed by Fukunaga and Narendra [2]. In the last years, some methods have been developed to improve the method proposed by Fukunaga and Narendra in two ways, the first one is focused on building a better tree that can lead to a faster classification process [3,4,5,6] and the second one is focused on the improvement of the search method [5,6,7,8].

The improvements proposed in [3,4,5,6,8] are exact methods to find the nearest neighbor and all of them are based on metric properties of the distance to avoid comparisons between objects. However, finding the nearest neighbor (even using a fast method) is a slow process for some tasks, therefore in [7] an approximated method is proposed, where Fukunaga's pruning rules are modified in order to finish the search when the current nearest neighbor is not too far from the exact nearest neighbor. In this process lower classification accuracy is obtained but in a shorter classification time.

It is important to highlight that until now, all methods based on tree structures were designed to work with numerical data when the comparison function satisfies metric properties.

### 2.1 The Classifier Proposed by Fukunaga and Narendra

In the first phase (*preprocessing phase*) of the method proposed by Fukunaga and Narendra [2], a tree structure is built as follows: the training set  $T$  is divided in  $l$  subsets, after that, each subset is divided into  $l$  subsets again (Fukunaga suggested using  $l=3$ ). This procedure is recursively applied to construct the tree, until certain level is reached. The  $K$ -Means [9] method is used for clustering the sets at each level in the tree, using the Euclidean distance.

Each node  $p$  of the tree contains four features, which are: the set of objects in the node  $p$  ( $S_p$ ), the number of objects in  $p$  ( $N_p$ ), the centre of the node ( $M_p$ ) and finally the maximum distance between the centre of  $p$  and the objects in the node  $p$  ( $R_p$ ).

Given a new object  $Q$  to classify, the Fukunaga fast  $NN$  classifier searches the nearest neighbor based on a branch and bound method to traverse the tree. Two pruning rules are used to decide whether a node or an object of the tree is evaluated or not. These rules are based on the triangle inequality. The first rule is applied to nodes of the tree and the second one is applied to objects that belong to leaves of the tree. For every child  $p$  of the current node, the pruning condition (first rule) is:

$$B + R_p < D(Q, M_p) \quad (1)$$

Where  $B$  is the distance between  $Q$  and the nearest object found so far and  $D$  is the comparison function. The nodes that satisfy this condition are not evaluated. In the final level of the tree, the objects that belong to that leaf node are tested to decide whether or not to compute the distance from the sample  $Q$  to the objects from the node. The pruning rule for every object  $o_i \in S_p$  is:

$$B + D(o_i, M_p) < D(Q, M_p) \quad (2)$$

The objects that satisfy condition (2) can not be closer than the nearest neighbor found so far and the distance to  $Q$  is not computed. The search process finishes when all nodes in the tree have been evaluated or eliminated by the pruning rule. Finally, the class of the nearest neighbor found in the search process is assigned to  $Q$ .

## 2.2 The Classifier Proposed by Moreno-Seco

The fast approximate  $NN$  classifier proposed by Moreno-Seco [7] creates a tree structure in the same way as it is proposed by Fukunaga, with  $l=2$  and only one object in each leaf node.

The Moreno-Seco fast  $NN$  classifier is based in the Fukunaga and Narendra search with a modification on the first pruning condition:

$$(1 + e)(D(Q, M_p) - R_p) > B \quad (3)$$

Where  $e$  is an error margin that allows to finish faster.

## 2.3 Comparison Functions for Mixed Data

Let us consider a set of objects  $\{O_1, O_2, \dots, O_N\}$ , each object described by  $d$  attributes  $\{x_1, x_2, \dots, x_d\}$ . Each attribute could be numerical or non numerical. To compare objects, in this work we used the function  $F$  used in [10,11] suitable to work with mixed data, which is defined as follows:

$$F(O_1, O_2) = \frac{|\{x_i \mid C_i(x_i(O_1), x_i(O_2)) = 1\}|}{d} \quad (4)$$

For qualitative data  $C_i(x_i(O_1), x_i(O_2))$  is defined as follows:

$$C_i(x_i(O_1), x_i(O_2)) = \begin{cases} 1 & \text{If } x_i(O_1) = x_i(O_2) \text{ and neither } x_i(O_1) \text{ nor } x_i(O_2) \\ & \text{is a missing value} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

For quantitative data  $C_i(x_i(O_1), x_i(O_2))$  is defined as follows:

$$C_i(x_i(O_1), x_i(O_2)) = \begin{cases} 1 & \text{If } |x_i(O_1) - x_i(O_2)| < \sigma_i \text{ and neither } x_i(O_1) \text{ nor } x_i(O_2) \\ & \text{is a missing value} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Where,  $\sigma_i$  is the standard deviation of the attributes  $x_i$ .

The function  $F$  allows comparing objects described by numerical and non numerical attributes even if there is missing information. Using the function  $F$ , the object that maximizes the function is considered as the most similar neighbor.

In this work we used the function  $S$  defined as follows.

$$S = (1 - F) \quad (7)$$

In this work, to compare objects we also used the functions: *HVDM*, described in [12] and *HOEM* [13], which also allow us to compare objects described by mixed data. Besides, in order to consider missing information the *HOEM* function was modified, using the same criterion that *HVDM* applies for missing values.

In the functions  $S$ , *HVDM* and *HOEM* the most similar neighbor is the object that minimizes the function.

### 3 Proposed Method

In this section, an approximate fast *MSN* classifier, based on a tree structure, which considers object described by mixed data is introduced. The method consists of two phases. The first phase, or preprocessing phase, is the construction of a tree structure from the set  $T$ , using suitable strategies for mixed data.

In the second phase, two fast approximate *MSN* search methods are used, which are independent of metric properties of the comparison function.

In order to compare the proposed methods, the search methods used in the second phase are compared against the search method proposed by Fukunaga [2] adapted for mixed data, which is an exact method based on metric properties. The proposed method is also compared with an adaptation for mixed data of the search method proposed by Moreno-Seco [7] which is an improvement over Fukunaga's method that allows to decrease the number of comparisons by introducing an error margin in the search. For this reason, Moreno-Seco's method is not an exact method anymore. However, this method also relies on metric properties to avoid comparisons.

There are other methods based in tree structures [14,15]. However, it is not possible to adapt these methods to work with similarity functions because these methods involve techniques such as *PCA* which are only applicable to numerical data.

#### 3.1 Preprocessing Phase

In this phase, it is proposed to create a tree structure from the training set. In order to build the tree structure, the training set is hierarchically decomposed. The *K-Means with Similarity Functions (KMSF)* algorithm, which is an extension of the *K-Means* algorithm suitable to work with mixed data [10,11] is used. In the original



*K-Means* the mean of the objects is considered as the centre of the cluster, meanwhile in *KMSF* an object of the cluster is used as the centre of it (see [10,11] for details).

The root of the tree (node 0) represents the whole set  $T$ . To create the first level of the tree, the set  $T$  is divided (using *KMSF*) in  $K$  disjoint sets (clusters), which form the  $K$  nodes of the tree in this level. Each node is divided again and this process is repeated until a stop criterion (SC) is satisfied. In figure 1 the algorithm to construct the tree is described.

Each node  $p$  of the tree contains three features which are: the set of objects that belong to the node  $S_p$ , the number of objects in the node  $N_p$  and unlike Fukunaga's and Moreno-Seco's methods a representative object of the node  $Rep_p$ .

---

```

NodesToDivide = Empty
CurrentNode = 0
Divide the objects from CurrentNode in K clusters
For every cluster  $c = 1:K$ 
    cluster( $c$ ) =  $node_c$  from the tree
     $node_c$  is a child of CurrentNode
    Compute the features of  $node_c$ :  $S_c, N_c, Rep_c$ 
    If  $N_c < 6$ 
         $node_c$  is a leaf
    Else
        Add  $node_c$  to the list NodesToDivide
End for every
 $p = K + 1$ 
While | NodesToDivide |  $\neq 0$ 
    CurrentNode = NodesToDivide [1]
    Subdivide the objects belonging from CurrentNode in K clusters
    For every cluster  $c = 1:K$ 
        cluster( $c$ ) =  $node_p$  from the tree
         $node_p$  is a child of CurrentNode
        Compute the features of  $node_p$ :  $S_p, N_p, Rep_p$ 
        If  $N_p < 6$ 
             $node_p$  is a leaf
        Else
            Add  $node_p$  to the list NodesToDivide
     $p = p + 1$ 
End for every
Eliminate CurrentNode from NodesToDivide
End while

```

---

**Fig. 1.** Tree building algorithm

### 3.2 Classification Phase

In this phase, two search methods to find an approximate most similar neighbor are used. In the first method (*MSN local search method*) we propose to use a modified depth-first search is used to find an approximate most similar neighbor and in the second method (*MSN global search method*) we propose to use a modified best-first search. In the original search methods (depth-first search and best-first search), an exhaustive search over all nodes of the tree is performed. To avoid the exhaustive tree traversal, the previous fast  $k$ -NN classifiers rely on pruning rules (based on metric properties). As we propose a method applicable when the comparison function does not satisfy metric properties and an exhaustive tree traversal would not be appropriate; then it is proposed to stop the search when a leaf of the tree is reached. The proposed methods for searching the *MSN* are described below:

*MSN local search method:* It begins at the root (node 0) of the tree following the path of the most similar node and finishing when a leaf of the tree is reached, where a local exhaustive search to find an approximate most similar neighbor  $O_{MSN}$  is performed (see pseudocode in figure 2).

---

```

Current Node = node 0
While Current Node ≠ Leaf
    Compute the similarity between Q and the representative objects from
    the descendant nodes of the Current Node.
    Select the most similar node to Q:
        MostSimilarNode =  $\min_{M_p \in \text{DesNode}} (HVD M (Q, Rep_p))$ 
    Current Node = MostSimilarNode
End while
Perform an exhaustive search with the objects in the Current Node
to report the  $O_{MSN}$  found so far:
     $O_{MSN} = \min_{O_i \in S_p} (HVD M (Q, O_i))$ 
Class(Q)=Class( $O_{MSN}$ )

```

---

**Fig. 2.** *MSN local search method*

*MSN global search method:* It begins at the root of the tree, comparing  $Q$  with the descendant nodes of the root, which are added to a list. After that, the list is ordered in such way that the most similar node to  $Q$  is in the first place. The most similar node is eliminated from the list and its descendant nodes are evaluated, added to the list and the list is ordered again. In this search it is possible to reconsider nodes in levels of the tree already traversed if the first node of the list belongs to a previous level in the tree (see pseudocode in figure 3).

The two search methods finish the search the first time they reach a leaf node, where an exhaustive search over the objects in the leaf node ( $S_p$ ) is performed to find the most similar neighbor  $O_{MSN}$ . After finding  $O_{MSN}$ , its class is assigned to the new object  $Q$ .

---

```

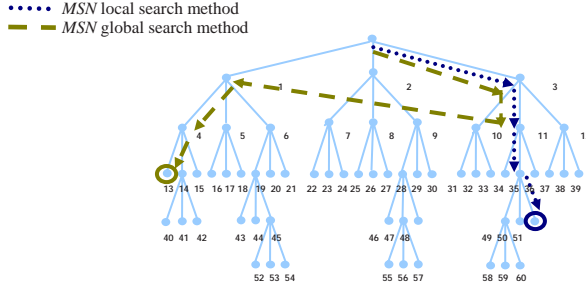
Current Node = node 0
List = {}
While Current Node ≠ Leaf
    Current Node = List [1]
    Delete Current Node from List
    Compute the similarity between Q and the representative objects of
    the descendant nodes of the Current Node.
    Add the nodes to List
    Order List in such way that the most similar object to Q is the first element of the list.
End while
Perform an exhaustive search with the objects in the Current Node
to report the  $O_{MSN}$  found so far:
     $O_{MSN} = \min_{O_i \in S_p} (HVD M (Q, O_i))$ 
Class(Q)=Class( $O_{MSN}$ )

```

---

**Fig. 3.** *MSN global search method*

In the figure 4 the difference between the search methods is shown. As we can see, global search method allows to evaluate nodes in levels already traversed.



**Fig. 4.** Traversals of the search methods

## 4 Experimental Results

In this section, we report the results obtained by applying the proposed fast approximate *MSN* classifier over 16 datasets from the *UCI* repository [16]. These datasets are numeric (Glass, Iris and Wine), non numeric (Tictac, Hayes, Soybean-large, Bridges and Mushrooms) and mixed (Hepatitis, Credit, Zoo, Flag, Echocardiogram, All-Hyper, Ann-thyroid and Thyroid 0387). In all the experiments 10-fold-cross-validation was used.

In the experimentation, the next five fast *NN* (*MSN*) classifiers were compared:

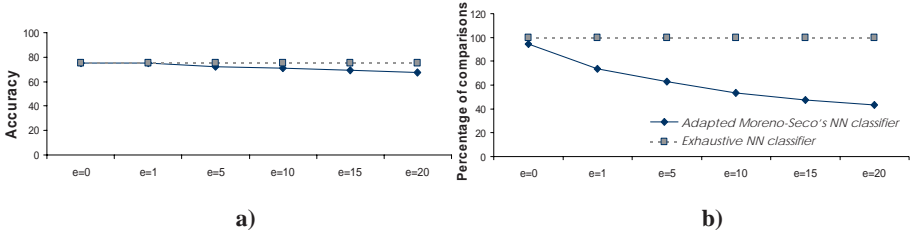
1. *The Exhaustive NN classifier* (using a dissimilarity function)
2. *Adapted Fukunaga's NN classifier*
3. *Adapted Moreno-Seco's NN classifier*
4. *The proposed method using MSN local search*
5. *The proposed method using MSN global search*

To compare *Fukunaga's* and *Moreno-Seco's classifiers* with our proposed methods, we adapted these classifiers. The adaptation consists on the use of the same tree structure proposed in section 3.1 and the same function suitable to work with mixed data, instead of a distance function.

In order to compare the classification methods, the accuracy and the number comparisons between objects are considered.

Before using *adapted Moreno-Seco's classifier*, some tests with different values of the parameter  $e$  were proved. To make this experiment the *HVDM* function was used and only Zoo, Hepatitis, Flag, Echocardiogram, Bridges, Glass, Iris and Wine were considered. From figure 5a we can see that at the time the parameter  $e$  grows, the error is incremented. But, the number of comparisons between objects is reduced (figure 5b). In the next experiments, *adapted Moreno-Seco's classifier* was used with  $e=20$ , because as we can see in figure 5, the number of comparisons is reduced with a slightly accuracy reduction.

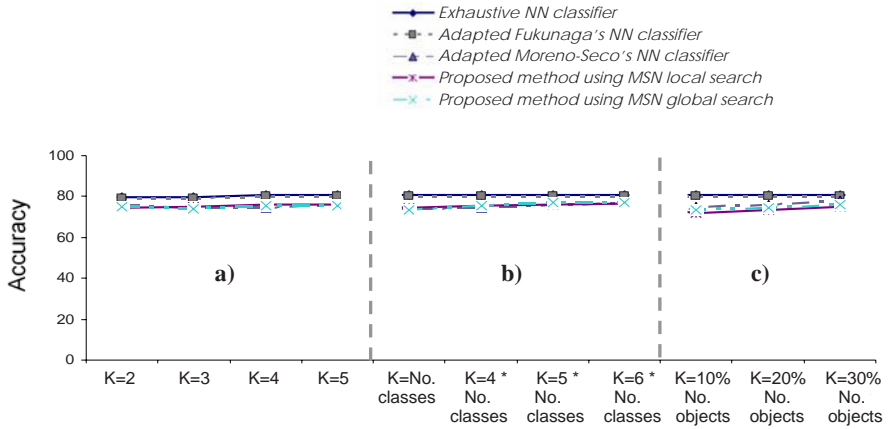
In order to evaluate the performance of the compared methods taking into account the size of the tree, some experiments were performed, using the datasets: Zoo,



**Fig. 5.** a) Accuracy performed by *adapted Moreno-Seco's classifier* with different values of  $e$ . b) Percentage of comparisons performed by *adapted Moreno-Seco's classifier* with different values of  $e$ .

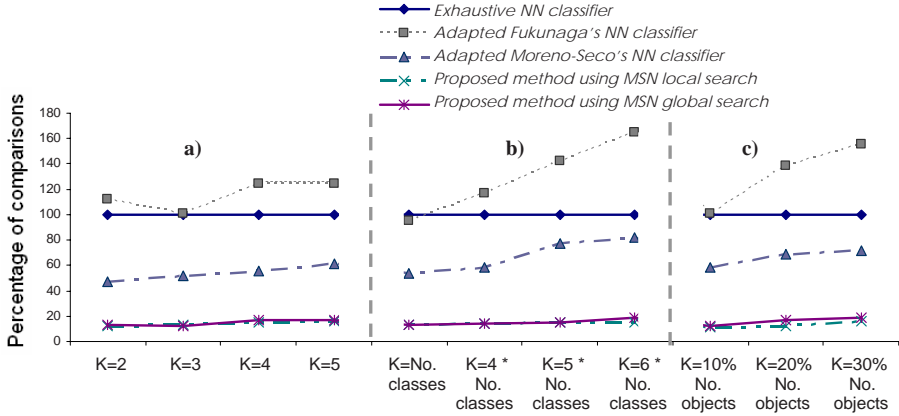
Hepatitis, Flag, Echocardiogram, Bridges, Glass, Iris and Wine, for selecting the value of  $K$  for the next experiments.

The parameter  $K$  of the *KMSF* algorithm corresponds to the number of branches of the nodes in the tree. The number of levels of the tree is not fixed, because it depends of the stop criterion (*SC*). So, in Figure 6 the accuracy for different values of  $K$  is shown. To evaluate the similarity between objects the *HVDM* function was used.



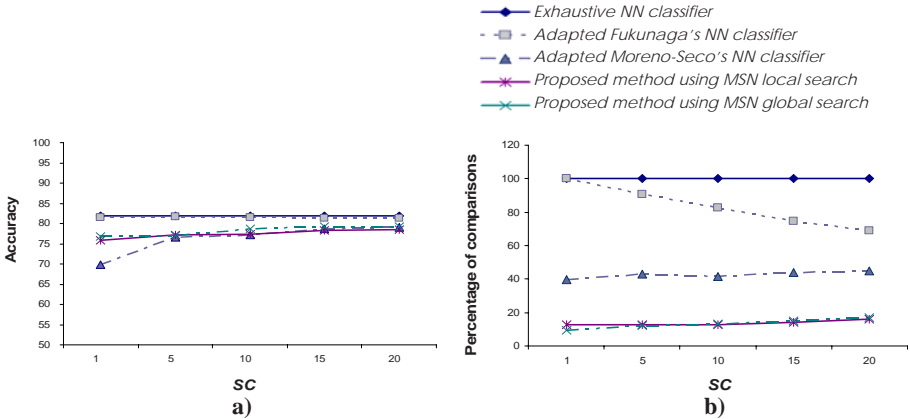
**Fig. 6.** Accuracy according to the values of  $K$  when: a)  $K \in [2, 5]$ , b)  $K$  is related to the number of classes (**No. classes**) in each dataset and c)  $K$  is related to the number of objects (**No. objects**) in each dataset.

In figure 7, the percentage of comparisons between objects for the same values of  $K$  is depicted, where the number of comparisons performed by the exhaustive search is considered as the 100 % of comparisons. According to figure 6, the accuracy increases when  $K$  grows. However, the number of comparison between objects (figure 7) increases too, in some cases it is even higher than 100%. In the next experiments  $K=3$  was used, because as we can see in figures 6 and 7 there is not a big variation of the accuracy and for  $K=3$  the number of comparisons is reduced.



**Fig. 7.** Percentage of comparisons between objects according to the values of  $K$  when: **a)**  $K \in [2, 5]$ , **b)**  $K$  is related to the number of classes (**No. classes**) in each dataset and **c)**  $K$  is related to the number of objects (**No. objects**) in each dataset.

Another important parameter in the tree algorithm is the stop criterion ( $SC$ ). In the preprocessing phase, each node of the tree is divided until there are few objects in a node. In the figure 8a, the accuracy obtained according different values of  $SC$  ( $SC = 1, 5, 10, 15$  and  $20$  objects) is depicted. The percentage of comparisons between objects is shown in figure 8b. From figures 8a and 8b it is possible to see that the accuracy and the percentage of comparisons do not vary too much using the methods proposed in this work.



**Fig. 8.** **a)** Accuracy according to different values of the stop criterion ( $SC$ ). **b)** Percentage of comparisons between objects according to different values of the stop criterion ( $SC$ ).

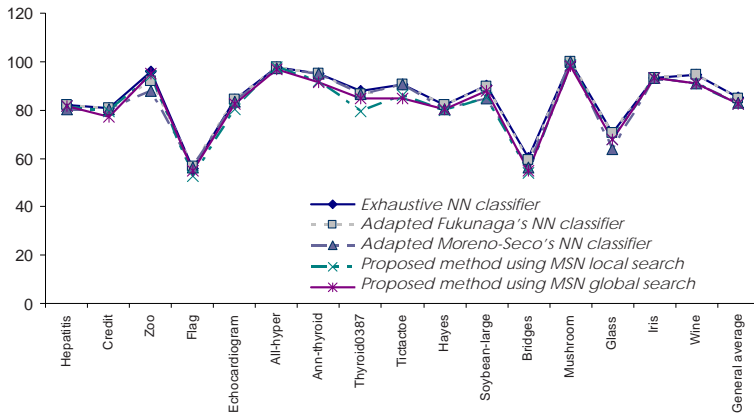
In table 1, the accuracy obtained (**Acc**) and the percentage of comparisons between objects (**Comp. Percen.**) are shown, using  $K=3$  and  $SC=20$ . The number of comparisons performed by exhaustive search is considered as the 100 % of comparisons.

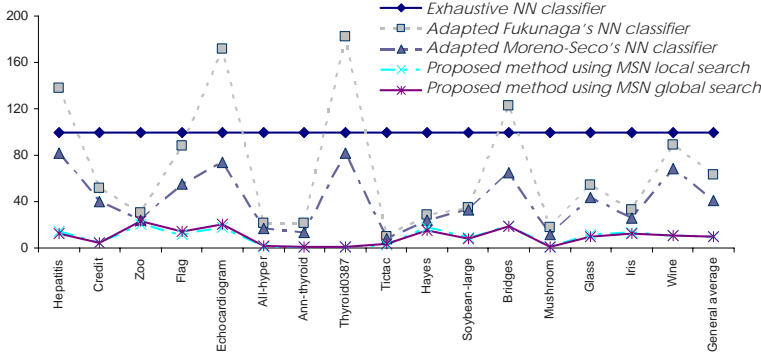
**Table 1.** Results using the *HVDM* function

Datasets	Exhaustive NN classifier		Adapted Fukunaga's NN classifier		Adapted Moreno-Seco's NN classifier		Proposed method using MSN local search		Proposed method using MSN global search	
	Acc	Comp. Percen.	Acc	Comp. Percen.	Acc	Comp. Percen.	Acc	Comp. Percen.	Acc	Comp. Percen.
Hepatitis	82,1	100	82,1	138,10	80,20	82,18	81,66	14,1	81,75	12,32
Credit	80,9	100	80,9	51,97	80,1	39,63	79,4	4,41	77,13	4,24
Zoo	96,0	100	92,09	30,14	88,09	23,68	94,2	20,25	95,09	23,48
Flag	55,6	100	56,6	88,41	56,13	55,12	52,7	11,34	54,73	13,90
Echocardiogram	84,1	100	84,1	171,16	83,46	73,7	80,16	18,02	82,4	20,24
All-hyper	97,9	100	97,9	20,92	97,21	16,82	97,5	1,22	96,8	1,36
Ann-thyroid	95,0	100	95,0	21,59	94,90	13,59	92,0	0,52	91,25	0,59
Thyroid0387	87,8	100	86,47	181,89	86,74	81,97	79,6	0,61	84,7	0,61
<b>Avg. of mixed Data</b>	<b>84,93</b>	<b>100</b>	<b>84,4</b>	<b>88,02</b>	<b>83,35</b>	<b>48,34</b>	<b>82,15</b>	<b>8,81</b>	<b>82,98</b>	<b>9,59</b>
Tictac	90,6	100	90,6	9,91	90,41	8,10	85,92	3,06	84,87	3,19
Hayes	82,0	100	82,0	28,43	80,16	24,23	80,05	17,67	80,1	14,67
Soybean-large	89,9	100	89,58	34,73	84,90	32,84	85,4	8,53	87,7	7,76
Bridges	60,4	100	59,18	122,76	56,36	65,02	54,09	18,96	54,77	18,84
Mushroom	100	100	100	18,16	99,95	11,22	98,74	0,45	98,1	0,50
<b>Avg. of non numerical data</b>	<b>84,58</b>	<b>100</b>	<b>84,27</b>	<b>42,80</b>	<b>82,36</b>	<b>28,28</b>	<b>80,84</b>	<b>9,73</b>	<b>81,11</b>	<b>8,99</b>
Glass	70,3	100	70,3	54,13	63,98	43,15	67,98	11,49	67,98	9,67
Iris	93,3	100	93,3	32,85	93,3	26,11	93,3	13,61	93,3	12,86
Wine	94,5	100	94,5	88,88	91,0	68,5	90,9	10,44	90,9	10,51
<b>Avg. of numerical data</b>	<b>86,03</b>	<b>100</b>	<b>86,03</b>	<b>58,62</b>	<b>82,76</b>	<b>45,92</b>	<b>84,06</b>	<b>11,85</b>	<b>84,06</b>	<b>11,01</b>
<b>General average</b>	<b>85,18</b>	<b>100</b>	<b>84,9</b>	<b>63,15</b>	<b>82,82</b>	<b>40,85</b>	<b>82,35</b>	<b>10,13</b>	<b>82,72</b>	<b>9,87</b>

The original *Fukunaga's NN classifier* is an exact method because of the triangle inequality property of the distance function. However, as the *HVDM* function does not necessarily satisfies this property, *adapted Fukunaga's NN classifier* (using the *HVDM* function) becomes an inexact method. As we can see from table 1, using the *adapted Fukunaga's NN classifier*, the accuracy does not decrease too much (from 85,18% to 84,9%). However, the number of comparisons between objects is only slightly reduced (from 100% to 63,15%).

Using *adapted Moreno-Seco's NN classifier* the percentage of comparisons is slightly reduced (40,85%). However, using the proposed method (with *MSN local* and *MSN global search*), the percentage of comparisons is much smaller (10,13% and 9,8%).

**Fig. 9.** Accuracy per dataset

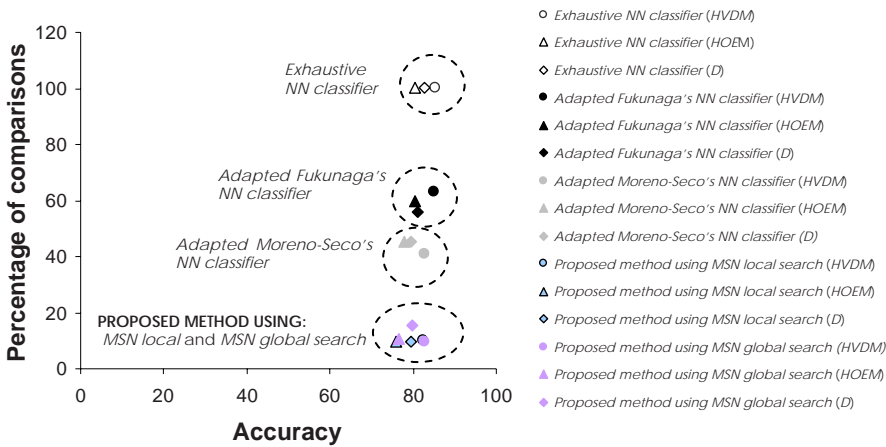


**Fig. 10.** Percentage of comparisons between objects per dataset

As it was mentioned in section 1, the fast *NN* classifier methods are proposed for problems where the set  $T$  is large; such is the case of the datasets: All-hyper, Ann-thyroid and Mushrooms, where a big reduction on the comparisons between objects was reached (less than 1%).

In figure 9, the accuracy obtained by the five fast classifiers listed before over each dataset is shown; here we can see that the performance of the five methods is very similar.

In figure 10, the number of comparisons between objects per dataset is depicted. It is possible to see from this figure that the number of comparisons using *adapted Fukunaga's classifier* sometimes is even higher than the exhaustive search. With *adapted Moreno-Seco's classifier*, the comparisons are always less than using *adapted Fukunaga's classifier*, but the proposed methods (*MSN local* and *MSN global search*) always did much less comparisons than *adapted Fukunaga's* and *Moreno-Seco's classifiers*.



**Fig. 11.** Accuracy against the comparisons percentage using the different classifiers and three different comparison functions

In figure 11 a graph of the accuracy against the comparisons percentage using the different fast *NN (MSN)* classifiers is shown. From this graph we can see that all the classifiers obtained similar accuracy but the proposed methods (using *MSN local* and *MSN global search*) did the smallest number of comparisons.

The fast *MSN* classifiers were also tested with *HOEM* and *S* functions. It is possible to see (from figure 11) that for different functions the accuracy and the number of comparisons do not vary too much among the classifiers.

## 5 Conclusions

In practical problems, it is frequent to find non numerical object descriptions or even mixed (numerical and non numerical). So, it is important to use suitable methods that allow us to work with these features.

According to our experimental results, it is possible to see that methods based on metric properties are not suitable when the objects are described by mixed attributes and the comparison function does not satisfy metric properties.

In this work, an approximated fast *MSN* classifier (with two search strategies) suitable for mixed data was proposed. In order to compare our method, *Fukunaga's* and *Moreno-Seco's classifiers* were implemented using the same comparison functions for mixed data. Based on our experimental results, unlike *adapted Fukunaga's* and *Moreno-Seco's classifiers*, our method (using *MSN local* and *MSN global search*), obtained a big reduction on the number of comparisons between objects with only a slightly accuracy reduction.

As future work we are going to look for an exact fast *MSN* classifier for mixed data, which could allow us to obtain the same accuracy that the exhaustive method, but reducing the number of comparisons.

## References

1. T. M. Cover and P. E. Hart. "Nearest neighbor pattern classification". Trans. Information Theory, vol. 13, pp. 21-27, 1967.
2. K. Fukunaga, P. Narendra, "A branch and bound algorithm for computing *k*-nearest neighbors". IEEE Trans. Comput. 24, pp. 743-750, 1975.
3. I. Kalantari and G. McDonald. "A data structure and an algorithm for the nearest point problem". IEEE Trans. Software Eng. 9, pp. 631-634, 1983.
4. S. Omachi and H. Aso. "A fast algorithm for a *k*-nn Classifier based on branch and bound method and computational quantity estimation". Systems and Computers in Japan, vol.31, no.6, pp.1-9, 2000.
5. E. Gómez-Ballester, L. Mico, J. Oncina. "Some Improvements in Tree Based Nearest Neighbor Search Algorithms". CIARP 2003, LNCS 2905, pp. 456-463, 2003.
6. E. Gómez-Ballester, L. Mico, J. Oncina. "Some approaches to improve tree-based nearest neighbor search algorithms". Pattern Recognition Letters 39 pp. 171-179, 2006.
7. F. Moreno-Seco, L. Mico and J. Oncina. "Approximate Nearest Neighbor Search with the Fukunaga and Narendra Algorithm and its Application to Chromosome Classification". CIARP, LNCS 2905, pp. 322-328, 2003.



8. L.Mico, J. Oncina and R. Carrasco. "A fast Branch and Bound nearest neighbor classifier in metric spaces". *Pattern Recognition Letters* 17, pp. 731-739, 1996.
9. J. B. MacQueen. "Some Methods for classification and Analysis of Multivariate Observations", *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, pp. 281-297, 1967.
10. J. R. García-Serrano and J. F. Martínez-Trinidad. "Extension to C-Means Algorithm for the use of Similarity Functions". *3<sup>rd</sup> European Conference on Principles and Practice of Knowledge Discovery in Database Proceedings*. Prague, Czech, pp. 354-359, 1999.
11. J. F. Martínez-Trinidad, J. R. García-Serrano and I. O. Ayaquica-Martínez. "C-Means Algorithm with Similarity Functions". *Computación y Sistemas*. Vol. 5, no. 4, pp. 241-246, 2002.
12. D. R. Wilson, T. Martínez. "Reduction techniques for instance based learning algorithms". *Machine Learning* .38, pp. 257-286, 2000.
13. D. Wilson, T. Martínez. "Improve heterogeneous Distance Functions". *Journal of Artificial Intelligence Research*, vol. 6, pp. 1-34, 1997.
14. J. McNames. "A Fast Nearest Neighbor Algorithm Based on a Principal Axis Search Tree". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 964-976, 2001.
15. C. Yong-Sheng, H. Yi-Ping and F. Chiou-Shann. "Fast and versatile algorithm for nearest neighbor search based on lower bound tree", *Pattern Recognition Letters*, 2006.
16. C. Blake, C. Merz. *UCI Repository of machine learning databases*. [<http://www.uci.edu/mlearn/databases/>], Department of Information and Computer Science, University of California, Irvine, CA, 1998.

# Performance Measures in Classification of Human Communications

Marina Sokolova and Guy Lapalme

Département d'informatique et de recherche opérationnelle  
Université de Montréal  
sokolovm@iro.umontreal.ca  
lapalme@iro.umontreal.ca

**Abstract.** This study emphasizes the importance of using appropriate measures in particular text classification settings. We focus on methods that evaluate how well a classifier performs. The effect of transformations on the confusion matrix are considered for eleven well-known and recently introduced classification measures. We analyze the measure's ability to retain its value under changes in a confusion matrix. We discuss benefits from the use of the invariant and non-invariant measures with respect to characteristics of data classes.

**Keywords:** Machine Learning, Evaluation Measures, Text Classification, Human Communication.

## 1 Introduction

Machine Learning has recently benefited from attention to the *performance measures* used in classification. The interest is supported by the development of new methods and their application in different domains. Evaluation of learning algorithms concentrates on two goals: comparison of algorithms and the applicability of algorithms on a specific domain. Empirical comparison is often done by applying algorithms on one or many data sets and then ranking the performance of the classifiers the algorithms have produced [1].

We focus on measures that evaluate how well a classifier identifies classes, without reference to computational costs or time. Specifically, we address the problem of performance measures for *new types of text classification*. The amount of web-posted texts necessarily invited applications of Data Mining (DM) and Text Data Mining (TDM), Machine Learning (ML) and Natural Language Processing (NLP) techniques. The data became a popular subject of DM, TDM, ML and NLP research through text classification (for a detailed review of the field refer to [2]). However, current studies on text classification undistinguish between texts in general and the data obtained in *human-to-human communication*. This led to leaving characteristics, specific to communications, out of the research scope. As a result, the same measures are used to evaluate classification performance on documents and on records of political debates, e.g., [3] and [4].

In this work we show that the problem of classification of communication records differs from the problem of general text classification. Thus, standard performance measures for text classification should be re-evaluated with respect to the characteristics of new problems. For this purpose we seek ways to compare various evaluation measures. We suggest a set of changes in a confusion matrix that correspond to specific characteristics of communication textual data. We analyze under what changes a measure retains its value, and therefore preserves the classifier's rank. This is called the measure's invariance under a change. We analyze *measure invariance* for several measures with respect to a transformation of a confusion matrix. Invariance properties identify measure applicability to particular learning settings. The presented analysis is supported by *examples of applications* where invariance properties of measures lead to good ranking of classifiers.

Establishing measure's invariance is one of the main goals of the measurement theory. If a measure is not invariant under the permissible transformations then statistical inference can be applied only to the measure values, but not to the measured attribute [5]. Data Mining has successfully exploited the invariant properties of interestingness measures for comparison of association and classification rules [6,7,8]. The invariant properties of classification measures recently had been discussed in [9], without specifically referencing them to text classification problems.

## 2 Human Communications in Text Classification

In recent years NLP and ML communities have turned their attention to studies of *opinions*, *subjective statements*, and *sentiments*. Data for these studies are found on chart-boards, blogs, product and movie reviews, and in email, records of phone conversations and political debates, electronic negotiations, etc. These sources represent records of *human communications*. Communication, through the variety of forms, conveys meanings sent by a speaker and received by a hearer. These meanings can be complex and subtly expressed and made up from what is said and what is implied [10].

Success of communication depends on the speaker's ability to produce a message and on the hearer's ability to understand it. Pragmatics, the study of language use, accepts that to be able to infer the meaning of the speaker's message, the hearer expects that the message satisfies standards of the Grice Maxims [11]: Quantity (informativeness), Quality (truthfulness), Relation (relevance) and Manner (clarity). These require the message to be as informative as the situation requires, trustworthy, relevant, clear and brief [10].

Not all communications satisfy Grice Maxims. Sometimes a hidden context interferes with the correct understanding of a message. We present examples of situations where communications and actions come in sharp contradiction. Seemingly successful negotiation given by Table 1 fails because the participant refuses to sign the agreement. Praise for a camera reported in Table 1 is also misleading because the user labels the camera as negative.

**Table 1.** Examples of situations where communicated meaning contradict actions of interlocutors

Communication	Action
<i>Dear XXX, I am YYY, a representative of Such and Such Company. Our company is interested in your [products] ... Dear XXX, I like your last offer and accept it. Thank you very much for your cooperation.</i>	The participant refuses to sign the agreement.
<i>A great camera! ... easy to use, viewfinder, flash are good ... it's a best buy!</i>	The user labels the camera as negative.

**Table 2.** Examples from communication categories

Means	Interaction types		
	one-to-one	one-to-few	one-to-many
Written	Letter	List email	Chart-board message
Verbal	Phone talk	Local radio announcement	Radio broadcast
Visual & verbal	Videophone talk	Video presentation	YouTube video
Face-to-face	Conversation	Lecture	Rally address

Language plays an important role in communication. The language role is critical in a situation when people communicate only verbally, e.g., by phone. In exclusively written communication language is the only tool to deliver a message. However, delivery of a message depends on many factors, including

- means, e.g., face-to-face meeting, email;
- topic of discussion, e.g., business, personal;
- time mode, i.e., synchronous or asynchronous;
- interaction mode, determined by the speaker-hearer ratio, e.g., one-to-one, one-to-many;
- speaker-hearer roles, e.g. doctor-patient, buyer-seller, presenter-audience; etc.

We suggest to use a two-dimensional Interaction-Means taxonomy that allows to distinguish between different types of interactions and mediations:

- one-to-many written: chart-boards, blogs, web-posted product and movie reviews;
- one-to-few face-to-face: political debates in the US Congress;
- one-to-few written: list email;
- one-to-one written: electronic negotiations.

Table 2 presents examples of different communication categories. Columns could be added with “few-to-one”, ..., “many-to-many” types.

Records of one-to-one and one-to-few communications, e.g., electronic negotiations and email discussions, are used in studies of individual behavior. The aim

of such studies is to find what factors influence behavior of a person in a specific situation. Classification of texts depends on the problem statement, e.g. [12,13]. Transcripts of the US Congress debates are used as a part of fast-growing studies of social networking. Here a common task is to define important influence factors and predict future behavior of members of a social group. In this case, records are classified according to actions of speakers, e.g. [4].

So far, records of one-to-many communications attract more attention and produce more volume of research than other types of communication. These records are studied as evaluative texts, i.e. delivering the author’s opinion on the discussed subject. Movie reviews, blogs are often used in sentiment analysis to find whether texts reflect positive or negative opinion of the author on certain products or events. In this case, texts are classified according to opinion/sentiment labels, e.g. [14,15]. Another popular learning task is to establish strength of the author’s opinion, e.g. [16].

### 3 Text Classification and Performance Measures

Quality of classification can be assessed using a confusion matrix, i.e., records of correctly and incorrectly recognized examples for each class. Table 3 reports on binary classification, where  $tp$  are true positive,  $fp$  – false positive,  $fn$  – false negative, and  $tn$  – true negative counts.

**Table 3.** A confusion matrix for binary classification

Class	Classified	
	as $pos$	as $neg$
$pos$	$tp$	$fn$
$neg$	$fp$	$tn$

In text classification, an input text needs to be classified into one (and only one) of  $j$  classes (or groups)  $C_1, \dots, C_j$ . The existence of the classes is known a priori. Work by Gabrilovich and Markovitch, e.g. [17], exemplifies characteristics of traditional text classification:

1. this is essentially classification of documents, e.g. research papers, technical reports, magazine articles, etc.
2. the main task is topic classification, e.g. identification of documents about Dallas, Texas, or documents about bands and artists, etc.
3. classes are built as relevant vs irrelevant documents, i.e., documents about Dallas, Texas, are distinguished from all other documents; hence, classes are built as positive vs “everything else”;
4. retrieval of relevant documents, or a positive class, is the most important task, thus focus is on  $tp$  classification.

Importance of retrieval of positive examples is reflected by the choice of performance measures for text classification:

$$Precision = \frac{tp}{tp + fp} \quad (1)$$

$$Recall = \frac{tp}{tp + fn} \quad (2)$$

$$Fscore = \frac{(\beta^2 + 1)tp}{(\beta^2 + 1)tp + \beta^2 fn + fp} \quad (3)$$

$$BreakEvenPoint = \frac{tp}{tp + fp} = \frac{tp}{tp + fn} \quad (4)$$

Three measures evaluate the classifier performance by calculating the ratio of correctly classified positive examples to examples labeled as positives (*Precision*), positive examples in data (*Recall*), and total positive examples, labeled and from data, (*Fscore*). *BreakEvenPoint* essentially estimates when disagreement between data and algorithm labeling of positive examples is balanced ( $fp = fn$ ). All these measures omit  $tn$  in their formulas, thus do not consider correct classification of negative examples.

Work by Lee *et al*, e.g., [4], concentrates on records of communications and presents direction in text classification started by [14] in 2002:

1. this is classification of political debates, web postings, phone calls, etc., i.e., records of human communications;
2. the main task is non-topic classification, e.g, vote classification, gender classification, mood classification, etc.
3. classes often have distinct features, e.g., male and female, success and failure, etc.; in this case positive and negative classes are both well-defined;
4. retrieval of a positive class, discrimination between classes, balance between retrieval of both classes are possible tasks whose importance depends on the problem at hand.

So far, there is no common understanding on the choice of measures used to evaluate performance of classifiers in opinion, subjectivity, and sentiment analysis. Employed performance measures are either

$$Accuracy = \frac{tp + tn}{tp + fn + fp + tn}, \quad (5)$$

which is used in [14,4] and other works by this group, or *Precision*, *Recall*, *Fscore*, e.g., [18], or correspondence between

$$Sensitivity = \frac{tp}{tp + fn} = Recall \quad (6)$$

and

$$Specificity = \frac{tn}{fp + tn} \quad (7)$$

reported in [13].

With different measures in use, it is important to know how performance evaluations, produced by those measures, relate to each other. Experimental evidence shows that disagreement happens quite often [13].

## 4 Invariance Properties of Measures

Finding appropriate measure is possible by establishing *how comparable are the involved measures*. Following [9], we focus on the ability of a measure to preserve its value under a change in a confusion matrix. The invariance of a measure signals that it does not detect this change. Depending on the learning goals, non detection can be beneficial or adverse.

For instance, text classification extensively uses *Precision* and *Recall* (*Sensitivity*). These measures do not detect changes in  $tn$ , when all other matrix entries remain the same. In document classification, a large number of unrelated documents constitutes a negative class that lacks unifying characteristics (a multi-modal negative class). The criterion for the performance of the classifier is its *performance on related documents* (a well-defined, unimodal, positive class) and may not depend on  $tn$ . *Precision* and *Recall* depend on  $tp$ , which shows agreement between data and algorithm labeling of positive examples, and  $fp$  and  $fn$ , which show disagreement between data and algorithm labeling of positive examples. Thus these measures provide the most important perspective on classifiers' performance for document classification. Another emerging application of text classification, classification of consumer reviews, works with highly related documents constituting unimodal positive and negative classes. Thus the evaluation measure may depend on classification of negative examples and reflect the  $tn$  change, when other matrix elements stay the same.

We examine the invariance properties with respect to basic changes of a matrix. Our claim is that the following invariance properties affect the measure's applicability and trustworthiness:

**Exchange of  $tp$  with  $tn$  and  $fn$  with  $fp$  (t1).** Table 4 shows the confusion matrix after the changes to the confusion matrix reported in Table 3. A measure is invariant if

$$m(tp, fn, tn, fp) = m(tn, fp, tp, fn) \quad (8)$$

This shows measure permanence with respect to classification results distribution. If the measure is invariant, then it does not distinguish  $tp$  from  $tn$  and  $fn$  from  $fp$  and may not recognize asymmetry of classification results. Thus it may not be trustworthy when classifiers are compared on data sets with different and/or unbalanced class distributions. For example, invariant measures may be more appropriate for assessment of classification of consumer reviews then for document classification.

**Change of true negative count (t2).** Table 5 presents the resulting confusion matrix. A measure is invariant if

$$m(tp, fn, tn, fp) = m(tp, fn, tn', fp) \quad (9)$$

This measure does not recognize specifying ability of classifiers. Such evaluation may be more applicable to domains with a multi-modal negative class, built as "everything not positive". If the measure is non-invariant, has  $\overline{t2}$ ,

then it acknowledges ability of classifiers correctly identify negative examples. If the measure is able to do this, it may be reliable for comparison in domains with a well-defined, unimodal, negative class. In case of text classification, these invariant measures are suitable for evaluation of document classification and non-invariant measures are preferable for evaluation of such communications where criteria exist for positive as well as for negative results.

**Change of a false count (t3).** Table 6 reports the confusion matrix. A measure is invariant if

$$m(tp, fn, tn, fp) = m(tp, fn, tn, fp') \quad (10)$$

t3 indicates measure constancy if disagreement increases between the data and classifier labels. An invariant measure shows preference for data labels. In case of unreliable data labeling such measure may give misleading results. A non-invariant measure may not be suitable for data with many counter examples. If classifier ranking improves when  $fp$  increases, the measure may favor a classifier prone to faux positives. In case of t3, the use of invariant and non-invariant measures might be decided based on problem and data characteristics. This is especially important for problems in sentiment classification of blogs, charts, consumer reviews, where some data do not have consistent labels because of the absence of rigorous labeling rules, and in classification of records of long-term communications, where some data have a substantial number of counter-examples.

**Classification scaling (t4).** Table 7 presents the confusion matrix. A measure is invariant if

$$m(tp, fn, tn, fp) = m(k_1 tp, k_2 fn, k_2 tn, k_1 fp) \quad (11)$$

This shows measure uniformity with respect to proportional changes of classification results. If the measure is non-invariant, then its applicability may depend on class sizes. If we expect that for different data sizes the same portion of examples exhibits positive (negative) characteristics, then the invariant measure may be a better choice for classifiers' evaluation. The non-invariant measures may be more reliable if we do not know how representative is the data sample in terms of proportion positive/negative examples (which is might be the case in web-posted consumer reviews).

## 5 Empirical Evidence

Application on “real life” communication data supports our claim on necessity of measure comparison. The data are the records of human-to-human electronic negotiations, where a buyer and a seller try to reach an agreement on virtual purchase of commercial products. A negotiation is successful when agreement is reached, otherwise it is failed. Support Vector Machine(SVM) and Naive Bayes (NB) have been applied on the same data set. Tables 8 and 9, adapted from



**Table 4.** Confusion matrix after the exchange of  $tp$  with  $tn$  and  $fn$  with  $fp$ 

Class	Classified	
	as $pos$	as $neg$
as $pos$	$tn$	$fp$
as $neg$	$fn$	$tp$

**Table 6.** Confusion matrix after a change in false positive count

Class	Classified	
	as $pos$	as $neg$
$pos$	$tp$	$fn$
$neg$	$fp'$	$tn$

**Table 8.** Confusion matrix for SVM

Class	Classified	
	as $pos$	as $neg$
$pos$	1242	189
$neg$	390	740

**Table 5.** Confusion matrix after a change in true negative count

Class	Classified	
	as $pos$	as $neg$
$pos$	$tp$	$fn$
$neg$	$fp$	$tn'$

**Table 7.** Confusion matrix after scaling

Class	Classified	
	as $pos$	as $neg$
$pos$	$k_1 tp$	$k_2 fn$
$neg$	$k_1 fp$	$k_2 tn$

**Table 9.** Confusion matrix for NB

Class	Classified	
	as $pos$	as $neg$
$pos$	1108	323
$neg$	272	858

[13], report their confusion matrices. The matrices are representative in a sense that changes in data representation do not statistically affect SVM and NB performance and, consequently,  $tp$ ,  $fn$ ,  $fp$ ,  $tn$ .

We apply several measures to rank the classifiers, starting with the listed in Section 3 evaluators. Except these evaluators, the employed measures include the Area Under Curve ( $AUC$ ), calculated for one run of the Receiver Operating Characteristic

$$AUC = \frac{1}{2} \left( \frac{tp}{tp + fn} + \frac{tn}{tn + fp} \right) \quad (12)$$

likelihoods  $\rho_+$ ,  $\rho_-$  that are frequently used for comparison of diagnostic tests [19],

$$\rho_+ = \frac{tp(tn + fp)}{fp(tp + fn)} \quad (13)$$

$$\rho_- = \frac{fn(tn + fp)}{tn(tp + fn)}. \quad (14)$$

Huang and Ling [20] newly introduced a combined measure that they denote as  $AUC:acc$

$$AUC:acc = \frac{Sensitivity + Specificity}{2 \cdot Accuracy} \quad (15)$$

First four columns of Table 10 report measure values and the classifier ranks. *Recall* is omitted because of co-linearity with *Sensitivity*. *Fscore* is used with

**Table 10.** Empirical comparison, invariance properties and clustering of performance measures

Measure	Empirical evidence				Invariance				Cluster		
	SVM		NB		t1	t2	t3	t4	1	2	3
	%	rank	%	rank							
<i>Accuracy</i>	77.4	1	76.8	2	+	-	-	-			✓
<i>Sensitivity</i>	86.8	1	77.5	2	-	+	+	-		✓	
<i>Specificity</i>	65.4	2	75.9	1	-	-	-	-	✓		
<i>Precision</i>	76.0	2	80.1	1	-	+	-	+		✓	
<i>Fscore</i>	81.2	1	78.9	2	-	+	-	-		✓	
<i>BreakEvenPoint</i>	74.0	1	72.4	2	-	+	-	-		✓	
<i>AUC</i>	52.3	2	53.4	1	-	-	-	-	✓		
$\rho_+$	2.51	2	3.22	1	-	-	-	-	✓		
$\rho_-$	0.20	1	0.30	2	-	-	-	-	✓		
<i>AUC:acc</i>	98.3	2	99.8	1	-	-	-	-	✓		

$\beta = 1$ . In four “Invariance” columns “+” and “-” denote invariance and non-invariance respectively on our data.

We also emphasize that measure’s focus on  $tp$  does allow to evaluate how well a classifier deals with the specific problems of human communication data (refer to examples given by Table 1). Most probably, those examples could be falsely classified as positives.

## 6 Analysis of Results

The invariant properties, introduced in Section 4, divide the measures into *three clusters*. One cluster is constructed from measures non-invariant under the four matrix transformations. *Specificity*, *AUC*,  $\rho_+$ ,  $\rho_-$  and *AUC:acc* change their values under all the considered changes in a confusion matrix.

- The first non-invariance,  $\overline{t1}$ , means that the measures are sensitive to asymmetry of classification. This is a well-known characteristic for *Specificity*, but not for the other four measures that have been recently introduced to classification. The non-invariance may explain why *AUC:acc* is more reliable than *Accuracy* when used for classifiers’ assessment on imbalanced data [20].
- The second non-invariance,  $\overline{t2}$ , signals that the use of the measures is more appropriate on data with a unimodal negative class than with a multi-modal one. This implication is more important for *AUC* and *AUC:acc* than for *Specificity* and  $\rho_+$ ,  $\rho_-$ . The latter are usually used in combinations with other measures, whereas the former might be applied separately.
- The third non-invariance,  $\overline{t3}$ , shows that the measures may be resistant to unreliable data labeling. To find out whether the measures may favor a classifier with a poor ability of detecting counterexamples, we have to check if ranking increases when  $fp$  increases. This is not true for ranking produced by

*Specificity*. Rankings produced by the other four measures are not monotonic under the property assumptions.

- The last non-invariance,  $t4$ , indicates that the measures may not be comparable when used on data with considerably different sizes.

The remaining five measures can be naturally categorized into *Accuracy* and the *Fscore* group (*Precision*, *Recall (Sensitivity)*, *Fscore* and *BreakEvenPoint*). All the *Fscore* group measures are invariant under the change of  $tn$ . This well-known property have made them a tool of choice for evaluations of document classification. Within this group, *Precision* is invariant under scaling, *Recall (Sensitivity)* – under the change of  $fp$  and *Fscore* and *BreakEvenPoint* have identical invariance properties ( $\overline{t1}$ ,  $t2$ ,  $\overline{t3}$ ,  $\overline{t4}$ ). The *Accuracy*'s only invariance,  $t1$ , has been much discussed in Machine Learning community. The last three columns of Table 10 represent three clusters containing the measures that correspond to the check marks ( $\checkmark$ ) in the lines.

*Invariance with respect to the matrix transformations* is especially important because it connects evaluation measures to particular learning settings. We summarize *applicability of measures* to subfields of text classification: document classification and classification of human communications. The initial assumption would be to apply *Fscore* measures as the most suitable for text classification evaluation. However, subfields' classification problems exhibit different characteristics. That may require applications of different evaluation measures. Based on the analysis of invariance properties of measures we propose the following:

- Document classification data are usually highly imbalanced. Relevant documents construct a small well-defined positive class, a populous negative class is built from non-relevant documents as “everything non-positive”. Presence of a multi-modal negative class favors the use of the *Fscore* measures.
- Classification of human communications often is mostly represented by sentiment classification where data are collections of free form texts of product evaluations. Proportion of positive and negative examples depends on the popularity of a product. Positive and negative classes are well-defined. Due to presence of a unimodal negative class, *Sensitivity* and *Specificity* may provide more reliable classifier ranking than *Precision* and *Recall (Sensitivity)*. *AUC:acc* may be preferable over *Accuracy* if there is a class imbalance. However, other measures might be suitable for classification of communications in social activities, such as political debates or electronic negotiations. If data have a unimodal negative class and a large number of counter examples, as in records of electronic negotiations, *Accuracy*, *Precision*, *Recall (Sensitivity)*, and *Specificity* may be used for reliable classification ranking.

## 7 Conclusions and Future Work

We have analyzed applicability of performance measures to different subfields of text classification. We have shown that document classification differs from classification of human communications, thus that these two types of text classification may require different set of performance measures.

We have shown that the results of the classifier comparison depend on a number of factors, including invariant properties of the measures. We have considered effects of various transformations of the confusion matrix on several well-known performance measures. The invariance properties have lead to fine distinctions of relations between the measures and the data characteristics. One way to insure reliable evaluation is to employ a measure corresponding to the learning setting. The next step would be to expand the list of connections between learning settings and evaluation measures.

This approach opens new directions for future work. First, we built a framework for the *two-dimensional* relations “measure vs invariance” and omitted decision theory relations. Note that the listed measures evaluate different decision aspects of the classifier performance. Given below is a condensed description from [19,1]:

- *Accuracy*, *Recall (Sensitivity)*, *Specificity* show how effectively a classifier identifies the data labels;
- *Precision* estimates the class agreement of the data labels with the labels given by the classifier;
- *AUC* indicates the classifier’s ability to avoid false classification;
- $\rho_+$  and  $\rho_-$  assess prediction ability on positive and negative classes respectively.

Combining the decision aspects with the existing framework leads to constructing a *three-dimensional* “measure vs invariance vs decision aspect” taxonomy of measures.

Next, this study focuses on binary classification. A natural way to extend it is to apply similar systematization to *multi-class classification*. Multi-class extension is desirable because many Machine Learning applications switch from binary “positive vs everything else” to finer grained problems. For example, in sentiment analysis, opinion mining and other subfields of subjectivity analysis three-class classification problems gradually substitute binary classification problems.

Next, our study concentrates on text classification, but it can be expanded to other language applications of Machine Learning. Machine Translation and Natural Language Processing are other examples of the fields where the discussed measures, e.g., *Fscore*, are used for comparison of classifiers.

**Acknowledgments.** This work has been funded by the Natural Sciences and Engineering Research Council of Canada.

## References

1. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* **7** (2006) 1–30
2. Sebastiani, F.: Machine learning in automated text categorization. *ACM Computing Surveys* **34**(1) (2002) 1–47

3. Koppel, M., Schler, J.: Authorship verification as a one-class classification problem. In: Proc 21st International Conf on Machine Learning ICML'04. (2004) 489–495
4. Thomas, M., Pang, B., Lee, L.: Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. (2006) 327–335
5. Sarle, W.S.: Measurement theory: Frequently asked questions. In: the Disseminations of the International Statistical Applications Institute. ACG Press (1996) 61–66
6. Geng, L., Hamilton, H.: Interestingness measures for data mining: A survey. *ACM Computing Surveys* **38**(3) (2006)
7. Lallich, S., Teytaud, O., Prudhomme, E.: Association rules interestingness: measure and validation. In F., G., J., H.H., eds.: *Quality Measures in Data Mining*. Springer (2006) –
8. Tan, P., Kumar, V., Srivastava, J.: Selecting the right objective measure for association analysis. *Information Systems* **29**(4) (2004) 293–313
9. Sokolova, M.: Assessing invariance properties of evaluation measures. In: Proceedings of the Workshop on Testing of Deployable Learning and Decision Systems, the 19th Neural Information Processing Systems Conference (NIPS 2006). (2006)
10. Leech, G.: *Principles of Pragmatics*. second edn. Longman (1991)
11. Grice, P.: *Studies in the Way of Words*. Harvard University Press (1989)
12. Boparai, J., Kay, J.: Supporting user task based conversations via email. In: Proc 7th Australasian Document Computing Symposium. (2002)
13. Sokolova, M.: *Learning from Communication Data: Language in Electronic Business Negotiations*. Ph.D. dissertation. (2006)
14. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? sentiment classification using machine learning techniques. In: Proc Empirical Methods of Natural Language Processing EMNLP'02. (2002) 79–86
15. Mishne, G.: Experiments with mood classification in blog posts. In: Proc 1st Workshop on Stylistic Analysis of Text for Information Access (Style2005). (2005) [staff.science.uva.nl/gilad/pubs/style2005-blogmoods.pdf](http://staff.science.uva.nl/gilad/pubs/style2005-blogmoods.pdf).
16. Wilson, T., Wiebe, J., Hwa, R.: Recognizing strong and weak opinion clauses. *Computational Intelligence* **22**(2) (2006) 7399
17. Gabrilovich, E., Markovitch, S.: Text categorization with many redundant features: Using aggressive feature selection to make svms competitive with c4.5. In: Proc 21st International Conf on Machine Learning ICML'04. (2004) 321–328
18. Gamon, M.: Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004). (2004) 841–847
19. Biggerstaff, B.: Comparing diagnostic tests: a simple graphic using likelihood ratios. *Statistics in Medicine* **19**(5) (2000) 649–663
20. Huang, J., Ling, C.: Constructing new and better evaluation measures for machine learning. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'2007). (2007) –

# Cost-Sensitive Decision Trees with Pre-pruning

Jun Du<sup>1</sup>, Zhihua Cai<sup>1,2</sup>, and Charles X. Ling<sup>2</sup>

<sup>1</sup> School of Computer Science

China University of Geosciences, Wuhan, 430074, P.R. China

<sup>2</sup> Dept. of Computer Science

The University of Western Ontario, London, Ontario, N6A 5B7, Canada

j.du@hotmail.com, zcai6@uwo.ca, cling@csd.uwo.ca

**Abstract.** This paper explores two simple and efficient pre-pruning strategies for the cost-sensitive decision tree algorithm to avoid over-fitting. One is to limit the cost-sensitive decision trees to a depth of two. The other is to prune the trees with a pre-specified threshold. Empirical study shows that, compared to the error-based tree algorithm C4.5 and several other cost-sensitive tree algorithms, the new cost-sensitive decision trees with pre-pruning are more efficient and perform well on most UCI data sets.

## 1 Introduction

For most previous research on classification, the main goal is to develop algorithms that minimize the number of errors on previously unseen examples. This is valid only when the costs of different errors are equal. In many real-world applications, however, it is far from the case. For example, in medical diagnosis, the errors for diagnosing someone as healthy carries a very high cost when that person in fact has a life-threatening disease, compared to the cost from mistakenly diagnosing a healthy one as having the disease. Cost sensitive classification deals with such cases where misclassification costs are not equal.

Generally, there are three main types of strategies for cost sensitive classification, implemented by manipulating one of the three components respectively: the train data, the learning algorithm, and the output of the learned model [1]. Many approaches have been developed in the past few years in making the traditional cost-insensitive classification algorithms cost-sensitive. For example, [2], [3], [4] discussed neural networks for cost-sensitive classification; [5] and [6] worked on cost-sensitive evolutionary algorithm; [7] made support vector machines sensitive to the cost; [8], [9] and [10] focused on the ensemble techniques such as bagging and boosting; decision tree algorithms, one of the most popular machine learning techniques, have also been studied for cost-sensitivity.

Current research in cost-sensitive decision trees falls into two categories. The first category is concerned with making the attribute splitting criterion sensitive to cost [11,12,13]. The other category develops new or modified pruning algorithms to minimize the expected cost [14,15]. In this paper, we propose two simple and efficient pre-pruning strategies for cost-sensitive decision

trees to avoid overfitting. Compared to the classical error-based tree algorithm C4.5 and several other cost-sensitive tree algorithms, our cost-sensitive decision trees with pre-pruning are more efficient and perform well on most UCI data sets.

The rest of the paper is organized as follows. In section 2, we first review unpruned cost-sensitive decision tree [13], which uses an attribute splitting criterion for reducing the total cost including the misclassification cost and test cost. Our new pre-pruning cost-sensitive trees are proposed after that. Then, we present our experiment results in section 3. Finally, section 4 draws conclusions and suggests future work.

## 2 Cost-Sensitive Decision Trees

### 2.1 Unpruned Cost Reduction Based Decision Tree

[13] proposes a new attribute splitting criterion for building cost-sensitive decision trees by minimizing the sum of misclassification and test cost [16]. In the decision tree building process, the algorithm directly chooses an attribute that reduces and minimizes the total cost (the sum of the misclassification cost and test cost) for the split, instead of choosing an attribute that minimizes the entropy (as in C4.5).

Similar to the traditional error-based decision tree algorithms, this cost minimization algorithm may have the deficiency of overfitting the training examples. That is, when a decision tree is built, some branches may be built reflecting anomalies in the train data due to noise or outliers, and this often leads to good performance on the train data but bad on test data. Pre-pruning and post-pruning are two typical methods to remove the least reliable branches and generally result in faster and better classification ability for independent test data.<sup>1</sup>

### 2.2 Cost-Sensitive Decision Trees with Pre-pruning

The algorithms we proposed in this paper are based on [13], incorporating two simple pre-pruning methods, described below.

**2-Level Tree.** With this approach, we just build the tree with no more than 2 levels. [18] and [19] have used similar approaches for error-based tree building, and shown that simpler trees often work quite well in many data sets. In this paper, we use the same idea in the cost-sensitive tree building process. The

---

<sup>1</sup> [13]’s work includes both misclassification costs and attribute costs. Attribute costs can act as a natural pruning mechanism, because an expensive attribute is unlikely to be chosen to split the data further, unless there is a large gain in the reduction of the misclassification cost. Nevertheless, overfitting could still happen, especially when the attribute cost is small or zero (as we study here). [17] incorporates post-pruning in cost-sensitive decision trees.

empirical study in section 3 will show that indeed the simple approach works quite well.

**Threshold Pruning Tree.** Another common approach for pre-pruning is imposing a pre-specified threshold on the splitting measure. Using cost reduction alone, the unpruned tree [13] would be expanded until the cost reduction is smaller than or equal to 0. We set a threshold on the cost reduction to avoid overfitting. We assume that the tree expansion is worthwhile only when the cost reduction is greater than the sum of False Positive(FP) and False Negative(FN) cost (we assume that the cost of True Positive and True Negative is 0). That is:

$$Threshold = FP + FN$$

For cost-sensitive trees with both pre-pruning methods, the following is used to label leaves. If the cost reduction is 0 or negative (for the 2-level trees), or if the cost reduction is less than the threshold (pre-specified threshold pruning), a leaf node is formed, and it should be labeled as the class minimizing the expected cost according to train data falling into the node. If no instance is falling into a node, then a leaf is also formed labeled as the class minimizing the expected cost of its parent node.

### 3 Empirical Study

#### 3.1 Configuration

We conduct experiments on the new algorithms above and compare them against the classical error-based algorithm C4.5 and cost-sensitive algorithms including

**Table 1.** UCI data sets used in the empirical study

Data set	No. of attributes	No. of examples	Class distribution
Breast-cancer	9	277	196/81
Breast-w	9	699	458/241
Colic	22	368	232/136
Credit-a	15	690	307/383
Credit-g	20	1000	700/300
Diabetes	8	768	500/268
Heart-statlog	13	270	150/120
Hepatitis	19	155	32/123
Ionosphere	34	351	126/225
Kv-vs-kp	36	3196	1669/1527
Labor	17	57	20/37
Sick	29	3772	3541/231
Sonar	60	208	97/111
Vote	16	435	267/168



the original unpruned cost reduction tree [13], weighting approach (with and without minimum expected cost) [20] and MetaCost [9], all implemented in Weka [21]. Only misclassification cost is considered in this paper (other types of cost will be studied in the future work).

14 data sets from UCI Machine Learning Repository [22] are used in the empirical study, all of which have discrete attributes and binary class without missing values. Information on these data sets is tabulated in Table 1.

Five cost matrices are used on these UCI data sets to evaluate the effects of different cost ratios. The costs of true positive and true negative are always set as 0, while false positive is always set to 1, and false negative cost is set to be 2, 5, 10, 20 and 50. This makes the cost ratio as 2, 5, 10, 20 and 50, which is supposed to show algorithms' performance with different cost ratios.

Under each cost matrix, 10-fold cross validation is performed on each data set. The experiment is repeated for 10 times and the average cost (total cost divided by the size of the test data) is recorded in the final results. Two-tailed t-test with a 95% confidence level is conducted to examine statistical significance.

### 3.2 Experiments Results

A total of 7 algorithms are compared; they are: pruned C4.5 labeled as “C4.5(P)”, unpruned cost reduction tree [13] labeled as “CR”, instance-weighting approach (with and without minimum expected cost) [20] with C4.5 labeled as “C4.5csmc” and “C4.5cs”, MetaCost [9] with C4.5 labeled as “MetaCost”, the proposed 2-level tree labeled as “CR-2”, and the proposed threshold pruning tree labeled as “CRPrune”. Table 2 lists the average misclassification cost, and Table 3 lists the corresponding summary on the t-test. Each entry  $w/t/l$  in Table ?? means that the algorithm at the corresponding row wins in  $w$  data sets, ties in  $t$  data sets, and loses in  $l$  data sets, compared to the algorithm at the corresponding column. The same notation is used in Table 5.

**Table 2.** Average misclassification cost on UCI data sets[illegible]

**Table 2.** (*Continued*)

Cost Ratio = 5							
Data set	C4.5(P)	CR	C4.5cs	C4.5cs-mc	MetaCost	CR-2	CRPrune
breast-cancer	0.110	0.094	0.083	0.076	0.074	0.068	0.068
breast-w	0.018	0.017	0.014	0.015	0.014	0.018	0.012
colic	0.058	0.050	0.050	0.052	0.053	0.048	0.045
credit-a	0.049	0.073	0.030	0.037	0.032	0.037	0.036
credit-g	0.099	0.087	0.062	0.068	0.065	0.060	0.063
diabetes	0.105	0.082	0.052	0.054	0.050	0.051	0.052
heart-statlog	0.066	0.106	0.051	0.054	0.052	0.065	0.063
hepatitis	0.047	0.060	0.021	0.026	0.025	0.026	0.021
ionosphere	0.022	0.027	0.012	0.019	0.015	0.019	0.015
kr-vs-kp	0.002	0.034	0.002	0.002	0.002	0.034	0.034
labor	0.027	0.041	0.023	0.014	0.018	0.016	0.019
sick	0.008	0.006	0.005	0.006	0.006	0.006	0.005
sonar	0.078	0.096	0.047	0.069	0.051	0.054	0.063
vote	0.011	0.009	0.009	0.010	0.010	0.009	0.009
Cost Ratio = 10							
Data set	C4.5(P)	CR	C4.5cs	C4.5cs-mc	MetaCost	CR-2	CRPrune
breast-cancer	0.216	0.181	0.072	0.075	0.071	0.069	0.071
breast-w	0.032	0.023	0.019	0.019	0.022	0.014	0.015
colic	0.112	0.139	0.076	0.076	0.068	0.081	0.080
credit-a	0.092	0.144	0.039	0.056	0.051	0.047	0.047
credit-g	0.189	0.099	0.068	0.088	0.073	0.074	0.070
diabetes	0.204	0.145	0.060	0.073	0.065	0.063	0.058
heart-statlog	0.123	0.197	0.064	0.076	0.065	0.070	0.057
hepatitis	0.083	0.108	0.021	0.026	0.021	0.034	0.021
ionosphere	0.033	0.048	0.025	0.025	0.016	0.022	0.014
kr-vs-kp	0.003	0.034	0.004	0.004	0.004	0.034	0.034
labor	0.042	0.072	0.035	0.016	0.021	0.019	0.035
sick	0.015	0.008	0.007	0.010	0.009	0.009	0.008
sonar	0.138	0.179	0.047	0.091	0.062	0.077	0.048
vote	0.020	0.015	0.019	0.017	0.015	0.016	0.015
Cost Ratio = 20							
Data set	C4.5(P)	CR	C4.5cs	C4.5cs-mc	MetaCost	CR-2	CRPrune
breast-cancer	0.427	0.346	0.071	0.080	0.071	0.084	0.077
breast-w	0.061	0.051	0.022	0.031	0.028	0.021	0.019
colic	0.218	0.262	0.066	0.085	0.064	0.088	0.070
credit-a	0.177	0.272	0.046	0.053	0.044	0.045	0.045
credit-g	0.368	0.137	0.074	0.104	0.070	0.075	0.070
diabetes	0.401	0.268	0.065	0.079	0.064	0.068	0.069
heart-statlog	0.237	0.365	0.056	0.100	0.060	0.082	0.056
hepatitis	0.154	0.204	0.021	0.032	0.021	0.052	0.021
ionosphere	0.056	0.087	0.025	0.039	0.018	0.027	0.018
kr-vs-kp	0.007	0.034	0.006	0.006	0.008	0.034	0.034
labor	0.072	0.135	0.035	0.019	0.019	0.026	0.035
sick	0.030	0.014	0.011	0.017	0.016	0.016	0.010
sonar	0.260	0.345	0.047	0.126	0.059	0.097	0.047
vote	0.039	0.029	0.025	0.031	0.027	0.030	0.027

Table 2. (Continued)

Cost Ratio = 50							
Data set	C4.5(P)	CR	C4.5cs	C4.5cs-mc	MetaCost	CR-2	CRPrune
breast-cancer	1.062	0.895	0.071	0.094	0.071	0.136	0.071
breast-w	0.149	0.125	0.031	0.053	0.031	0.030	0.030
colic	0.538	0.632	0.063	0.121	0.063	0.133	0.063
credit-a	0.432	0.646	0.044	0.067	0.044	0.050	0.044
credit-g	0.906	0.255	0.070	0.159	0.070	0.082	0.070
diabetes	0.993	0.635	0.065	0.106	0.069	0.080	0.065
heart-statlog	0.579	0.882	0.056	0.172	0.063	0.121	0.056
hepatitis	0.367	0.490	0.021	0.050	0.021	0.104	0.021
ionosphere	0.126	0.206	0.036	0.080	0.019	0.044	0.018
kr-vs-kp	0.017	0.034	0.010	0.011	0.011	0.034	0.034
labor	0.161	0.325	0.035	0.030	0.019	0.047	0.035
sick	0.075	0.033	0.014	0.039	0.036	0.038	0.011
sonar	0.625	0.831	0.047	0.248	0.047	0.177	0.047
vote	0.095	0.073	0.057	0.107	0.083	0.088	0.080

Table 3. Summary of the t-test on average misclassification cost

C R		C4.5(P)	CR	C4.5cs	C4.5cs-mc	MetaCost
2	CR-2	5/5/4	7/4/3	4/3/7	5/4/5	2/6/6
	CRPrune	6/4/4	9/4/1	7/0/7	5/4/5	5/3/6
5	CR-2	10/3/1	9/5/0	4/2/8	6/5/3	4/5/5
	CRPrune	12/1/1	12/2/0	3/6/5	8/3/3	7/3/4
10	CR-2	13/0/1	11/2/1	4/3/7	8/5/1	4/5/5
	CRPrune	12/1/1	11/3/0	4/7/3	10/2/2	7/4/3
20	CR-2	13/0/1	11/2/1	0/5/9	8/4/2	1/3/10
	CRPrune	13/0/1	13/1/0	3/8/3	11/1/2	3/6/5
50	CR-2	12/1/1	11/1/2	0/3/11	8/3/3	0/3/11
	CRPrune	12/1/1	12/2/0	2/10/2	12/1/1	2/10/2

Table 4 lists the average model training time (on a PC with Intel P4 3.0G Hz CPU and 512M memory), and Table 5 lists the corresponding summary with the t-test. As the cost ratio does not affect the model training time, we take only one cost ratio (cost ratio = 10) for the result.

Table 4. Average model training time on UCI data sets

Data set	C4.5(P)	CR	C4.5cs	C4.5cs-mc	MetaCost	CR-2	CRPrune
breast-cancer	0.004	0.005	0.004	0.005	0.046	0.003	0.002
breast-w	0.004	0.011	0.004	0.005	0.056	0.005	0.006
colic	0.011	0.013	0.011	0.010	0.095	0.004	0.005
credit-a	0.017	0.027	0.015	0.013	0.146	0.007	0.004
credit-g	0.043	0.017	0.043	0.038	0.378	0.008	0.006
diabetes	0.013	0.017	0.016	0.013	0.133	0.004	0.005
heart-statlog	0.006	0.008	0.007	0.005	0.062	0.002	0.001

**Table 4.** (*Continued*)

hepatitis	0.003	0.003	0.003	0.003	0.032	0.004	0.001
ionosphere	0.009	0.016	0.013	0.010	0.104	0.007	0.009
kr-vs-kp	0.117	0.065	0.095	0.120	1.260	0.070	0.065
labor	0.002	0.001	0.001	0.001	0.013	0.001	0.000
sick	0.057	0.105	0.066	0.061	0.630	0.075	0.096
sonar	0.011	0.025	0.012	0.012	0.129	0.009	0.003
vote	0.005	0.005	0.005	0.005	0.057	0.003	0.004

**Table 5.** Summary of the t-test on model training time

	C4.5(P)	CR	C4.5cs	C4.5cs-mc	MetaCost
CR-2	8/5/1	9/5/0	8/6/0	8/5/1	14/0/0
CRPrune	10/3/1	11/3/0	10/3/1	10/3/1	14/0/0

From the experiment results, several interesting observations can be made:

First, when the cost ratio is rather low (cost ratio = 2), CR-2 and CRPrune perform better than CR: the  $w/t/l$  value on the average misclassification cost is 7/4/3 between CR-2 and CR, and 9/4/1 between CRPrune and CR. However, they do not outperform other algorithms. In fact, no algorithm wins all the time on all data sets; even cost-based algorithms do not always perform significantly better than error-based algorithms. The low cost ratio makes the task similar to error-based classification, where cost-sensitive approaches have no particular advantage.

Second, when the cost ratio is high (cost ratio  $\geq 5$ ), the proposed algorithms significantly outperform C4.5, CR and even C4.5cs-mc. In addition, CRPrune are comparable to C4.5cs and MetaCost, while CR-2 performs worse than them.

Third, the proposed new algorithms CR-2 and CRPrune are the definite winner on the model training time compared with all other algorithms. Simplicity and efficiency are certainly significant advantages of the proposed algorithms.

## 4 Conclusions and Future Work

This paper explores two simple and efficient pre-pruning strategies for the cost-sensitive decision tree algorithm to avoid overfitting. One is to limit the cost-sensitive decision trees to a depth of two. The other is to prune the trees with a pre-specified threshold. Empirical study shows that, compared to the error-based tree algorithm C4.5 and several other cost-sensitive tree algorithms, our cost-sensitive decision trees with pre-pruning are more efficient and perform well on most UCI data sets.

In the future, we plan to incorporate other pruning methods in our algorithms. In addition, it is also valuable to extend our pre-pruning cost-sensitive trees to include other types of cost.

## References

1. Margineantu, D.: Methods for Cost-Sensitive Learning. PhD thesis, Oregon State University
2. Kukar, M., Kononenko, I.: Cost-sensitive learning with neural networks. In: Proceedings of the 13th European Conference on Artificial Intelligence. (1998) 445–449
3. Wan, C., Wang, L., Ting, K.: Introducing cost-sensitive neural networks. In: Proceedings of the 2nd International Conference on Information, Communications and Signal Processing, Singapore (1999) 445–449
4. Zhou, Z.H., Liu, X.Y.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering* **18**(1) (2006) 63–77
5. Turney, P.: Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research* **2** (1995) 369–409
6. Kwedlo, W., Kretowski, M.: An evolutionary algorithm for cost-sensitive decision rule learning. In: Proceedings of the 12th European Conference on Machine Learning. (2001) 288–299
7. Peng, Y., Huang, Q., Jiang, P., Jiang, J.: Cost-sensitive ensemble of support vector machines for effective detection of microcalcification in breast cancer diagnosis. *Lecture Notes in Computer Science* **3614** (2005) 483–493
8. Ting, K., Zheng, Z.: Boosting trees for cost-sensitive classifications. In: Proceedings of the 10th European Conference on Machine Learning, Germany (1998) 191–195
9. Domingos, P.: Metacost: A general method for making classifiers cost-sensitive. In: Proceedings of the 15th International Conference on Knowledge Discovery and Data Mining, ACM Press (1999) 155–164
10. Fan, W., Stolfo, S., Zhang, J., Chan, P.: Adacost: Misclassification cost-sensitive boosting. In: Proceedings of the 16th International Conference on Machine Learning. (1999) 97–105
11. Drummond, C., Holte, R.: Exploiting the cost (in) sensitivity of decision tree splitting criteria. In: Proceedings of the 17th International Conference on Machine Learning. (2000) 239–246
12. Ferri, C., Flach, P., Orallo, J.: Learning decision trees using the area under the roc curve. In: Proceedings of the 19th International Conference on Machine Learning. (2002) 139–146
13. Ling, C., Yang, Q., Wang, J., Zhang, S.: Decision trees with minimal costs. In: Proceedings of the 21st International Conference on Machine Learning. (2004)
14. Bradley, A., Lovell, B.: Cost-sensitive decision tree pruning: Use of the roc curve. In: Proceedings of the 18th Australian Joint Conference on Artificial Intelligence, Australia (1995) 1–8
15. Bradford, J., Kunz, C., Kohavi, R., Brunk, C., Brodley, C.: Pruning decision trees with misclassification costs. In: Proceedings of the European Conference on Machine Learning. (1998) 131–136
16. Turney, P.: Types of cost in inductive concept learning. In: Proceedings of the Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning. (2000)
17. Ling, C., Sheng, V., Bruckhaus, T., Madhavji, N.: Maximum profit mining and its application in software development. In: Proceedings of The 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06) (Industrial Applications Track). (2006) 929–934

18. Holte, R.: Very simple classification rules perform well on most commonly used datasets. *Machine Learning* **11** (1993) 63–91
19. Auer, P., Holte, R., Maass, W.: Theory and applications of agnostic pac-learning with small decision trees. In: *Proceedings of the 12th International Conference on Machine Learning*, Morgan Kaufmann (1995) 21–29
20. Ting, K.: Inducing cost-sensitive trees via instance weighting. In: *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery*, Springer-Verlag (1998) 23–26
21. Witten, I., Frank, E., eds.: *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann (2005)
22. Blake, C., Keogh, E., Merz, C.: *Uci repository of machine learning databases*. Department of Information and Computer Science, University of California (1998) <http://www.ics.uci.edu/~mllearn/MLRepository.html>.

# Probability Based Metrics for Locally Weighted Naive Bayes

Bin Wang and Harry Zhang

Faculty of Computer Science, University of New Brunswick  
P.O. Box 4400, Fredericton, NB, Canada E3B 5A3  
{bin.wang,hzhang}@unb.ca

**Abstract.** Locally weighted naive Bayes (LWNB) is a successful instance-based classifier, which first finds the neighbors of the test instance using Euclidean metric, and then builds a naive Bayes model in the local neighborhood. However, Euclidean metric is not the best choice for LWNB. For nominal attributes, Euclidean metric has to order and number the values of attributes, or judge whether the attribute values are identical or not. For numeric attributes, Euclidean metric is not appropriate for different attribute scales and variability, and encounters the problem of attribute value outliers when normalizing values. In this paper, we systematically study probability based metrics, such as Interpolated Value Difference Metric (IVDM), Extended Short and Fukunaga Metric (SF2), SF2 calibrated by logarithm (SF2LOG) and Minimum Risk Metric (MRM), and apply them to LWNB. These probability based metrics can solve the above problems of Euclidean metric since they depend on the difference between the probabilities to evaluate the distances between the instances. We conduct the experiments to compare the performances of LWNB classifiers using Euclidean metric and probability based metrics on UCI datasets. The results show that LWNB classifiers using IVDM outperform the ones using Euclidean metric and other probability based metrics. We also observe that SF2, SF2LOG and MRM do not perform well due to their inaccurate probability estimates. An artificial dataset is built by logical sampling in a Bayesian network, where accurate probability estimates can be produced. We conduct the experiment on the artificial dataset. The results show that SF2, SF2LOG and MRM using accurate probability estimates perform better than Euclidean metric and IVDM in LWNB.

**Keywords:** LWNB, Probability Based Metrics, IVDM, SF2, SF2LOG, MRM.

## 1 Introduction

In practice, Bayes' theorem is intractable since there are no sufficient training instances to obtain an accurate estimate of the full joint probability distribution. Naive Bayes classifier is a very popular and successful application of Bayes' theorem, which makes an extreme assumption by assuming that all the attributes are

conditionally independent given the class. It is well-known that this assumption is rarely held in the real world. But the performance and efficiency of naive Bayes surprises the researchers in many classification problems. Various approaches have been developed to improve the performance of naive Bayes. Many of them believe that, if the naive conditional independent assumption can be reduced, naive Bayes can be improved without impacting its simplicity and computational efficiency. Locally weighted naive Bayes (LWNB) [1] is one of the most successful approaches. LWNB is an instances-based learning method which simply stores the training instances in the training time, and then defers building a classifier until the testing time. When a test instance arrives, a naive Bayes classifier is built using a set of weighted training instances in the locale of the test instance. Because there are fewer training instances needed to build a local naive Bayes model, there is less chance of encountering strong dependencies in the neighborhood, and the effects of attribute dependencies are relieved. The experimental results show that LWNB improves classification accuracy dramatically compared with naive Bayes classifier and other related classifiers.

Like other instances-based learning methods, it is very important for LWNB to consider which metric should be selected to calculate the distance between the instances. Frank et al. [1] use Euclidean metric to evaluate the instances' distances. Although Euclidean metric proves the effect, it still has some problems. First, Euclidean metric is not a natural way to tackle the nominal attributes. It has to deal with the nominal attributes by two means: one is to order and number the values of the nominal attributes and to treat them as numeric values; the other is to judge whether the values are identical or not. Second, Euclidean metric is not always suitable for different scales (e.g. mm of Hg vs. degree Celsius) and different variability (e.g. high variability of blood pressure vs. low variability of body temperature) of numeric attributes. A simple way to fix the problem is to normalize numeric attribute values. But it is not satisfying when an outlier appears (e.g. body temperature = 50). [14]

In this paper, we systematically study probability based metrics, such as Interpolated Value Difference Metric (IVDM) [2], Extended Short and Fukunaga Metric (SF2) [3], SF2 calibrated by logarithm (SF2LOG), and Minimum Risk Metric (MRM) [3], and apply them into LWNB. IVDM is an augmented VDM [4] which not only uniformly deals with both nominal and numeric attributes as the heterogeneous distance metric, but also uses interpolation to alleviate the discretization problems. SF2 is extended from Short and Fukunaga Metric [5] which is proposed to minimize the expected value of the difference between the misclassification error with finite samples and the one with infinite samples. We design SF2LOG based on SF2, which uses logarithm to calibrate the probabilities. Compared with SF2, MRM directly minimizes the finite misclassification risk and relies on simpler optimality condition. In general, these probability based metrics evaluate the distances between the instances by the difference between the probabilities. They can solve the problems of Euclidean metric. For nominal attributes, the probabilities are estimated by counting the frequencies of occurrences, and there is no need to order and number the values,



or judge whether they are identical or not. For numeric attributes, the probabilities can be produced by the standard Gaussian distribution. Normalization for different scales and different variability is not necessary, and it is certain that there is no worry about the outliers. We conduct experiments to compare the performances of LWNB classifiers using Euclidean metric and probability based metrics on UCI datasets. From the experimental results, LWNB classifiers using IVDM outperform the ones using Euclidean metric and other probability based metrics. We also observe that SF2, SF2LOG and MRM do not perform well due to inaccurate probability estimates. Furthermore, an artificial dataset is designed by logical sampling [16] in a Bayesian network, in which the accurate class membership probabilities can be computed. We conduct the experiment on the artificial dataset. The results show that SF2, SF2LOG and MRM with accurate probability estimates perform better than Euclidean metric and IVDM in LWNB.

The rest of paper is organized as follows. In Section 2, we describe the procedure of LWNB. In Section 3, various probability based metrics are introduced. In Section 4, we present the experiments for LWNB classifiers using different metrics on UCI datasets and the artificial dataset, and also analyze and discuss the experimental results. Finally, we summarize our work and bring forward the future work in Section 5.

## 2 Locally Weighted Naive Bayes

The idea of LWNB is similar to locally weighted linear regression [6] which is considered as a local likelihood method from the statistical perspective. A local naive Bayes model is built within the neighborhood of the test instance. The training instances in this neighborhood are weighted by distances. The farther from the test instance, the less weights are assigned to the training instances. The final classification result is obtained from the local naive Bayes model.

At the beginning, the distance between each training instance and the test instance is computed.  $k$  is a user-specified parameter which controls how many instances are used to form the neighborhood of the test instance. The array of the distances is sorted ascendingly and the  $k$ th distance is defined as bandwidth.

Let  $d_i$  be the distance between the test instance and the  $i$ th nearest neighbor  $x_i$ , and  $d_k$  be the bandwidth.  $d_i$  is scaled by  $d_i/d_k$ . After that, some of distances are between zero and one if their un-scaled distances are equal to or smaller than  $d_k$ , whereas the other distances are greater than one. Then a monotonically decreasing weighted kernel is chosen to assign the weights. In our experiments, we use the linear weighting function which lets the weight decrease linearly with the distance. It is defined as  $w_i = f_{linear}(d_i) = \text{Max}(0, 1 - d_i)$ , where  $d_i$  is a scaled distance. This function means that the instances whose un-scaled distances equal bandwidth are assigned weight zero; some of instances, which are farther away from the test instance, are also assigned weight zero; the instances whose scaled distances between zero and one receive non-zero weights. The instances with non-zero weights are kept to form the neighborhood of the test instance.

Frank et al. [1] found that the performance of LWNB can be improved empirically by further rescaling the weights so that the total of the instances' weights is approximately  $k$ . Assume that there are  $m$  training instances  $x_i$  used to generate the local naive Bayes. Then the rescaled weights  $w'_i$  are computed as  $w'_i = w_i * m / \sum_{q=1}^n w_q$ , where  $n$  is the total number of training instances.

### 3 Probability Based Metrics

#### 3.1 Interpolated Value Difference Metric

The Value Difference Metric (VDM) was introduced by Stanfill and Waltz [4] as a valuable distance function for nominal attributes. A simplified version of the VDM defines the distance between two attribute values  $a_i$  and  $a_j$  of an attribute  $A$  as

$$vdm_A(a_i, a_j) = \sum_{q=1}^C |P(c_q|a_i) - P(c_q|a_j)|, \quad (1)$$

where  $C$  is the number of class labels,  $P(c_q|a_i)$  and  $P(c_q|a_j)$  are nominal attribute conditional probabilities which are estimated by computing frequencies of attribute values' occurrences.

Wilson and Martinez [2] extend VDM to a set of heterogeneous metrics which can deal with a dataset having both nominal and numeric attributes. Interpolated Value Difference Metric (IVDM) is an excellent one in VDM family. IVDM can be applied directly to numeric attributes which alleviates the need for normalization between attributes.

In IVDM, the values of numeric attributes are discretized into  $s$  equal-width intervals, where  $s$  is an integer supplied by users. In this paper,  $s$  takes the value of  $C$  or 5, whichever is greater, where  $C$  is the number of class labels. The width  $w_A$  of a discretized interval for attribute  $A$  is given by:

$$w_A = |max_A - min_A|/s, \quad (2)$$

where  $max_A$  and  $min_A$  are the maximum and minimum values, respectively, occurring in the training instances for  $A$ . The discretized value  $u$  of an attribute value  $a_i$  is given by:

$$u = discretize_A(a_i) = \begin{cases} a_i & \text{if } A \text{ is discrete, else} \\ s & \text{if } a_i = max_A, \text{ else} \\ \lfloor (a_i - min_A)/w_A \rfloor + 1 \end{cases} \quad (3)$$

The difference between IVDM and other VDMs is that IVDM needs to retain the original numeric values after the discretization. This can be useful to distinguish the numeric values which fall into the same interval after the discretization. IVDM assumes that attribute conditional probabilities hold true values at the midpoint of each interval, and the interpolation between the midpoints is helpful

to produce better probability estimates. Given two instance  $x$  and  $y$ , the distance function for IVDM is defined as:

$$IVDM(x, y) = \sum_{j=1}^m ivdm_{A^j}(a_x^j, a_y^j)^2, \quad (4)$$

where  $m$  is the number of attributes of instance  $x$  and  $y$ ,  $A^j$  is the  $j$ th attribute,  $a_x^j$  and  $a_y^j$  are the attribute values of  $A^j$  in  $x$  and  $y$  respectively, and  $ivdm_{A^j}(a_x^j, a_y^j)$  is defined as:

$$ivdm_{A^j}(a_x^j, a_y^j) = \begin{cases} vdm_{A^j}(a_x^j, a_y^j) & \text{if } A^j \text{ is discrete} \\ \sum_{q=1}^C |P'(c_q|a_x^j) - P'(c_q|a_y^j)| & \text{otherwise} \end{cases} \quad (5)$$

where  $P'(c_q|a_x^j)$  and  $P'(c_q|a_y^j)$  is the interpolated probabilities of numeric attribute values. In the general case, we assume that  $a$  is an attribute value of numeric attribute  $A$ ,  $u$  is an integer where  $u = discretize_A(a)$ ,  $c$  is a class label. The interpolated probability  $P'(c|a)$  is defined as:

$$P'(c|a) = P(c|u) + \left( \frac{a - mid_{a,u}}{mid_{a,u+1} - mid_{a,u}} \right) * (P(c|u+1) - P(c|u)). \quad (6)$$

In Equation 6,  $mid_{a,u}$  and  $mid_{a,u+1}$  are midpoints of two consecutive discretized ranges such that  $mid_{a,u} \leq a \leq mid_{a,u+1}$ .  $P(c|u)$  is the nominal attribute conditional probability which is estimated by frequencies of occurrences in the range  $u$  (and similarly for  $P(c|u+1)$ ). If  $x < mid_{a,u}$ ,  $u$  subtracts 1. Moreover, if  $u < 1$  or  $u > s$ ,  $P(c|u)$  is taken to be 0. The value of  $mid_{a,u}$  can be found as follows:

$$mid_{a,u} = min_A + w_A * (u + 0.5). \quad (7)$$

### 3.2 Short and Fukunaga Metric

Short and Fukunaga [5] address the problem of selecting the best distance measure for minimizing the difference between the finite sample nearest neighbor classification error and the asymptotic nearest neighbor error based on probabilities. In their work, they consider the metric for the 1-nearest neighbor and two-class case. Let  $x$  and  $y$  be two instances,  $c_1$  and  $c_2$  be two class labels, and  $P(c_1|x)$ ,  $P(c_1|y)$ ,  $P(c_2|x)$  and  $P(c_2|y)$  be the class membership probabilities. Then  $r(x, y) = P(c_1|x)P(c_2|y) + P(c_2|x)P(c_1|y)$  is the finite 1-nearest neighbor error rate, in which the probability of misclassifying  $x$  is evaluated by a particular metric given the nearest neighbor  $y$ .  $r^*(x) = 2P(c_1|x)P(c_2|x)$  is the asymptotic 1-nearest neighbor error rate [8]. Short and Fukunaga [5] show that minimizing the expectation  $E[(r(x, y) - r^*(x))^2]$  is equivalent to minimizing  $E[(P(c_1|x) - P(c_1|y))^2]$ , so the proper local metric is

$$SF(x, y) = |P(c_1|x) - P(c_1|y)|. \quad (8)$$

Myles and Hand [9] extend this metric to the multiple-class case and propose the following equation:

$$SF2(x, y) = \sum_{i=1}^C |P(c_i|x) - P(c_i|y)|. \quad (9)$$

In Equations 8 and 9, the class membership probabilities are estimated by naive Bayes estimator [15]. Although the computation of naive Bayes is effective, the produced probabilities is not accurate enough. We use logarithm to calibrate the probabilities to improve the performance. It is called SF2LOG which is shown as follows:

$$SF2LOG(x, y) = \sum_{i=1}^C |\log(P(c_i|x)) - \log(P(c_i|y))|. \quad (10)$$

### 3.3 Minimum Risk Metric

Blanzieri and Ricci propose the Minimum Risk Metric (MRM) [3], which is a very simple metric that directly minimizes the risk of misclassification.

Assume that there is an instance  $x$  and its nearest neighbor  $y$ . Given class label  $c_i$ , the finite risk of misclassifying  $x$  is  $P(c_i|x)(1 - P(c_i|y))$ . In the multiple-class case, the total finite risk is the sum of the risks extended to all the different classes, which is given by  $r(x, y) = \sum_{i=1}^C P(c_i|x)(1 - P(c_i|y))$ , where  $C$  is the number of class labels. Compared with the above approach of SF2 which minimizes the expectation of difference between the finite error and the asymptotic error, MRM directly minimizes the risk  $r(x, y)$  which is given as follows:

$$MRM(x, y) = r(x, y) = \sum_{i=1}^C P(c_i|x)(1 - P(c_i|y)). \quad (11)$$

In Equation 11, the class membership probabilities are also estimated by naive Bayes estimator [15].

## 4 Experimental Results

### 4.1 Experiments on UCI Datasets

In this section, we conduct the experiments on 33 UCI datasets [11] to compare the performances of LWNB classifiers using Euclidean metric (EUM), IVDM, SF2, SF2LOG and MRM. These UCI datasets consist of nominal attributes, or numeric attributes, or both of nominal and numeric attributes (for convenience, we call them mixture attributes). They are downloaded in the format of *arff* from the website of Weka [10]. We use the implementation of LWNB in Weka,

and implement IVDM, SF2, SF2LOG and MRM in Weka’s framework. Some typical values of  $k$  have been selected for LWNB classifiers. The influence of  $k$  will be described in the following section.

First, the experiments are conducted on discretized UCI datasets. We apply the filter of ReplaceMissingValues in Weka to replace the missing values of attributes. Then, we used the filter of Discretize, the unsupervised ten-bin discretization in Weka, to discretize numeric attributes. Thus, all the attributes are treated as nominal. It is well-known that, if the number of values of an attribute is almost equal to the number of instances in a dataset, this attribute does not contribute any information to classification. So we use the filter of Remove in Weka to delete such attributes. In 33 UCI datasets, Hospital Number in dataset colic, Instance Name in dataset splice and Animal in dataset zoo are deleted. We conduct the following experiments to compare the performances of LWNB classifiers using different metrics in terms of classification accuracy. The accuracy of each classifier is measured via the ten-fold cross validation for all datasets. Runs with the various algorithms are carried out on the same training sets and evaluated on the same test sets. The cross-validation folds are the same for all the experiments on each dataset. Furthermore, we conduct two-tailed t-test with a 95% confidence level to compare each pair of algorithms.

Next, we conduct experiments for LWNB classifiers using different metrics on UCI datasets which consist of numeric or mixture attributes. There are 25 such datasets in 33 UCI datasets. Similarly as above, we apply the ReplaceMissing-Values filter in Weka, and Remove filter which removes the attribute Animal in dataset zoo. The accuracy of each classifier is also measured via the ten-fold

**Table 1.** Experimental results on accuracy with discretized UCI datasets

Dataset	IVDM		EUM		SF2LOG		MRM
	K=30	K=50	K=30	K=50	K=50	K=50	
anneal	99.09	98.92	98.80	98.84	99.01	97.45	97.35
audiology	77.91	77.30	78.06	78.11	74.47	75.26	74.64
autos	78.82	77.94	79.65	78.57	78.35	74.05	73.71
balance-scale	88.59	89.23	85.92	85.92	95.27	97.06	97.15
breast-cancer	72.11	71.98	73.02	72.85	70.35	70.64	70.64
breast-w	97.07	97.21	96.61	96.74	96.07	95.80	95.80
colic	83.15	82.85	81.82	79.78	81.07	81.41	81.41
credit-a	86.06	86.09	85.57	86.13	84.35	84.36	84.36
credit-g	75.18	75.35	73.22	73.36	74.70	74.79	74.79
diabetes	74.82	75.31	72.82	72.95	73.42	73.58	73.58
glass	63.79	63.35	63.14	64.82	65.65	61.35	59.13
heart-c	81.88	81.62	80.40	80.46	80.83	80.85	80.62
heart-h	82.90	82.94	83.00	82.76	81.11	80.89	80.89
heart-statlog	83.00	82.56	80.89	81.07	82.96	83.26	83.26
hepatitis	83.56	84.64	85.28	85.98	84.91	85.21	85.21
hypothyroid	93.39	93.58	92.99	93.31	93.34	92.95	92.88
ionosphere	91.22	91.23	91.51	92.31	83.10	79.14	79.14
iris	96.00	96.00	94.47	95.33	95.87	95.40	95.40
kr-vs-kp	97.74	97.65	97.69	97.78	92.39	92.41	92.41
labor	93.10	95.10	93.90	94.20	95.47	96.50	96.50
lymph	85.77	84.68	84.54	84.12	83.30	82.73	82.06
mushroom	100.00	100.00	100.0	100.00	100.00	100.00	100.00
primary-tumor	45.13	46.79	43.77	45.66	44.63	45.58	44.66
segment	95.42	95.54	95.11	95.15	94.83	93.10	92.59
sick	98.06	98.08	98.08	98.05	97.45	97.44	97.44
sonar	83.04	82.31	83.28	82.52	72.69	72.98	72.98
soybean	93.57	93.66	92.93	93.48	92.87	92.97	92.83
splice	94.88	95.18	92.45	94.11	96.15	96.15	96.09
vehicle	71.80	71.17	71.10	71.63	70.39	66.06	64.88
vote	95.05	95.49	95.08	95.95	95.33	95.08	95.08
vowel	95.34	95.07	94.92	94.97	91.48	83.81	79.02
waveform-5000	82.66	82.94	76.70	78.38	83.30	82.70	82.70
zoo	96.94	95.75	96.25	96.25	96.05	96.15	96.05

**Table 2.** Summary of comparisons on discretized UCI datasets

	IVDM50	EUM30	EUM50	SF2LOG	SF2	MRM
IVDM30	0/33/0	0/25/8	0/27/6	2/26/5	2/22/9	2/22/9
IVDM50		0/27/6	0/28/5	2/25/6	2/21/10	2/21/10
EUM30			2/30/1	3/25/5	3/22/8	3/22/8
EUM50				3/25/5	3/22/8	3/22/8
SF2LOG					1/26/6	1/24/8
SF2						0/31/2

**Table 3.** Experimental results on accuracy with UCI datasets which consist of numeric or mixture attributes

dataset	IVDM	IVDM	EUM	EUM	SF2LOG	SF2	MRM
	K=30	K=50	K=30	K=50	K=50	K=50	K=50
anneal	98.96	99.14	98.58	98.33	97.18	98.27	98.27
autos	76.96	77.52	76.47	77.51	74.97	71.71	71.71
balance-scale	79.42	80.95	89.41	89.91	95.27	95.99	95.99
breast-cancer-w	96.07	95.62	96.28	96.31	94.99	95.65	95.65
coll.	82.31	80.06	78.61	79.76	79.39	79.14	79.14
credit-a	84.80	83.57	85.33	83.33	78.32	77.86	77.86
credit-g	73.11	73.51	74.29	75.07	72.03	72.41	72.41
diabetes	70.79	71.01	71.54	70.40	70.04	70.17	70.17
glass	74.54	73.96	71.31	72.20	67.75	66.22	66.22
heart-c	81.03	80.61	82.65	81.43	79.50	79.23	79.23
heart-h	80.04	82.18	82.38	82.53	81.76	81.97	81.97
heart-statlog	79.07	79.74	80.89	79.33	78.00	79.22	79.22
hepatitis	85.24	83.55	82.78	83.00	84.37	85.53	85.53
hypothyroid	96.79	97.00	95.49	96.29	96.16	96.87	96.61
ionosphere	91.43	89.49	80.23	82.91	92.74	91.83	91.83
iris	94.67	95.33	95.33	96.67	96.67	92.00	92.00
labor	95.00	93.33	91.67	93.33	89.67	88.00	88.00
lymph	82.33	82.33	85.67	83.00	81.00	79.67	80.33
segment	96.58	96.67	97.45	96.54	93.07	92.16	91.34
sick	97.56	97.45	96.71	96.85	97.03	97.03	97.03
sonar	87.05	86.10	88.52	88.00	85.14	78.88	78.88
vehicle	73.87	74.70	75.42	75.30	71.63	70.33	68.92
vowel	98.38	98.48	96.26	96.87	96.77	92.73	90.20
waveform	82.82	81.68	81.52	81.82	83.36	83.86	84.36
zoo	99.00	98.09	98.00	98.00	97.00	98.00	98.00

cross validation for all datasets, and two-tailed t-test with a 95% confidence level is conducted to compare each pair of algorithms.

Table 1 and Table 3 display the accuracy of each classifier on two kinds of UCI datasets. The two-tailed t-test results are shown in Table 2 and Table 4. Each entry of Table 2 and Table 4 has the format of  $w/t/l$ . It means that, compared with the classifier using the metric in the corresponding row, the classifier using the metric in the corresponding column wins in  $w$  datasets, ties in  $t$  datasets and loses in  $l$  datasets.

From the experimental results, we can see that LWNB classifiers using IVDM (particularly when  $k = 30$ ) outperform LWNB classifiers using Euclidean metric and other probability based metrics. Compared with LWNB classifiers using Euclidean metric (the one when  $k = 50$  is the best one reported by [1]), LWNB classifiers using IVDM improve the performances on two kinds of UCI datasets. SF2 and MRM do not perform well on discretized UCI datasets. On UCI datasets consisting of numeric and mixture attributes, however, the performances of SF2 and MRM are close to Euclidean metric. SF2LOG performs better than SF2 and MRM on two kinds UCI datasets.

**Table 4.** Summary of comparisons on UCI datasets which consist of numeric or mixture attributes

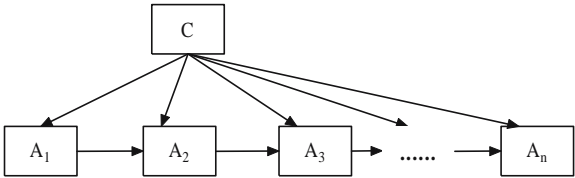
	IVDM50	EUM30	EUM50	SF2LOG	SF250	MRM50
IVDM30	1/23/1	1/16/8	1/16/8	1/17/7	1/18/6	1/19/5
IVDM50		2/15/8	2/18/5	1/18/6	2/18/5	2/18/5
EUM30			2/21/2	3/18/4	4/16/5	4/16/5
EUM50				2/19/4	3/17/5	3/17/5
SF2LOG50					0/23/2	0/22/3
SF250						0/25/0

The experimental results are not surprising. IVDM is based on the attribute conditional probabilities which are estimated directly by the frequencies of occurrences. However, SF2, SF2LOG and MRM have to use the estimates of the class membership probabilities by naive Bayes estimator. The estimates are constrained by the conditional attribute independent assumption so that the probabilities are not accurate. The inaccurate probability estimates hurt the performances of these metrics. By calibrating the probability estimates using logarithm, the performance of SF2LOG is better than SF2 and MRM.

**4.2 Artificial Dataset**

In this section, we design an artificial dataset from which the accurate probabilities can be produced. On this artificial dataset, we conduct an experiment to see the performances of SF2, SF2LOG and MRM using the accurate probabilities.

The artificial dataset is constructed by a Bayesian network. The structure of Bayesian network is defined as Figure 1, where the class variable node  $C$  is the parent of all the other nodes,  $A_1$  is the only node which has one parent, the other nodes  $A_2, \dots, A_n$  has two parents which are  $C$  and the preceding node. In this experiment, a network which has 25 nodes has been built, where node  $C$  represents the binary class label and other nodes represent the binary attributes. The conditional probability table is randomly generated on each node. We draw the dataset at random by logical sampling [16] in this Bayesian network. Given an instance from the artificial dataset, the accurate class membership probability of this instance is easy to compute by searching the conditional probability table on each node. The results of LWNB classifiers with Euclidean metric and



**Fig. 1.** The structure of the Bayesian network

**Table 5.** Experimental results on accuracy with artificial dataset

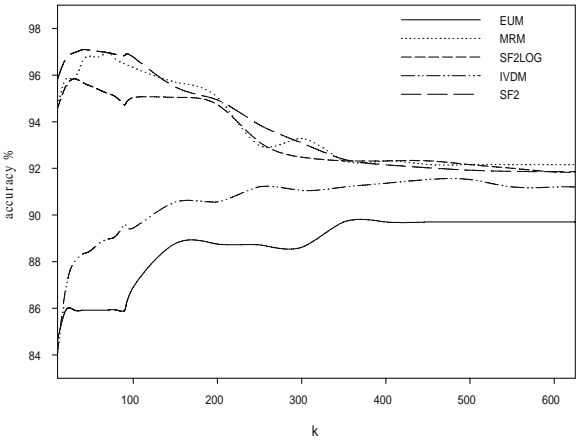
	EUM	IVDM	SF2	MRM	SF2LOG
Artificial dataset	92.80	93.80	94.20	94.70	96.00

probability based metrics are shown in Table 5, where  $k=100$ . In the experiment, SF2, SF2LOG and MRM use the accurate probability estimates computed from the Bayesian network. From the results, we can see that SF2, SF2LOG and MRM perform better than Euclidean metric and IVDM. Although it is not easy to get the accurate class membership probabilities in real-world applications, we can treat the accurate probabilities from this artificial dataset as a yardstick. When the probabilities estimates are more accurate, the probability based metrics, such as SF2, SF2LOG and MRM perform better for LWNB classifier.

### 4.3 Influence of $k$

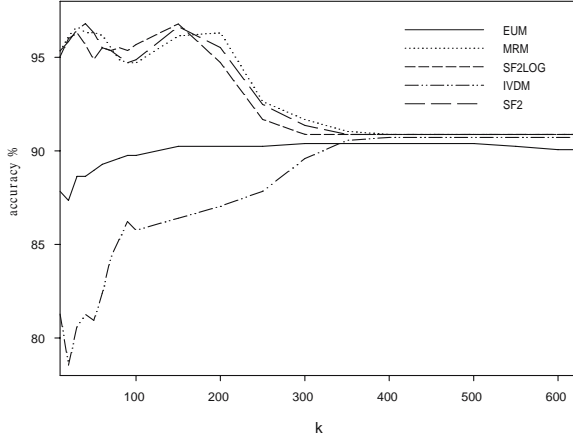
We conduct a series of experiments for LWNB classifiers using different metrics with various values of  $k$ . These experiments are conducted on one of UCI datasets, balance-scale, whose attributes are all numeric. The performances of LWNB classifiers using different metrics on the discretized and the original dataset with various values of  $k$  are shown in Figure 2 and Figure 3.

From the Figure 2 and Figure 3, we can see that the performance of LWNB classifier using Euclidean metric is not sensitive to the change of  $k$ . LWNB classifier using IVDM is not sensitive on discretized dataset. The performances of other probability based metrics swing within a small range before  $k < 300$ , and then stabilize.



**Fig. 2.** Performances of LWNB classifiers using different metrics on discretized balance-scale dataset





**Fig. 3.** Performances of LWNB classifiers using different metrics on original balance-scale dataset

## 5 Conclusion and Future Work

In this paper, we systematically study the probability based metrics, such as IVDM, SF2, SF2LOG and MRM, and apply them into LWNB classifiers. LWNB is a very successful instance-based learning classifier. Frank et al. use Euclidean metric on LWNB classifier. However, Euclidean metric has some problems when used to deal with nominal attributes or numeric attributes. Probability based metrics can solve these problems. They depend on probability estimates to evaluate the difference between the instances. Specifically, IVDM uses the sum of the difference between the attribute conditional probabilities which are estimated by frequencies of occurrences. SF2, SF2LOG and MRM utilize the class membership probabilities which are produced by naive Bayes estimator. We conduct the experiments for LWNB classifiers using Euclidean metric and probability based metrics on the discretized UCI datasets and UCI datasets which consist of numeric and mixture attributes. From the experimental results, IVDM outperforms other metrics on two kinds of UCI datasets. Due to the poor class membership probabilities estimated by naive Bayes estimator, SF2, SF2LOG and MRM do not perform well. But SF2LOG can improve the performance by calibrating probabilities using logarithm compared with SF2 and MRM. We also conduct the experiments on an artificial dataset which is built by logical sampling in a Bayesian network. Accurate probability estimates can be produced from the Bayesian network. The experimental results show that SF2, SF2LOG and MRM can perform better using more accurate probability estimates.

The main problem of probability based metrics is the estimation of probability. In the future work, we will try the probability calibration methods to calibrate the class membership probability. Zadrozny and Elkan [12] obtain the calibrated probability estimates from decision trees and naive Bayes classifier. They also

use pair-adjacent violators (PAV) [13] to calibrate probabilities. All the methods are worth trying to improve the performance of probability based metrics.

## References

1. Frank, E., Hall, M., Pfahringer, B.: Locally Weighted Naive Bayes. Proceedings of the Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann( 2003), 249-256.
2. Wilson, R. D., Martinez, T. R.: Improved Heterogeneous Distance Functions. Journal of Artificial Intelligence Research, Vol. 6. (1997) 1-34.
3. Blanzieri, E., Ricci, F.: Probability Based Metrics for Nearest Neighbor Classification and Case-Based Reasoning. Lecture Notes in Computer Science, Vol. 1650. (1999) 14.
4. Stanfill, C., Waltz, D.: Toward Memory-based reasoning. Communication of the ACM, Vol. 29. (1986) 1213-1228.
5. Short, R. D., Fukunaga, K.: The Optimal Distance Measure for Nearest Neighbour Classification. IEEE Transactions on Information Theory, Vol. 27. (1981) 622-627.
6. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag. (2001).
7. Loader, C.: Local Regression and Likelihood. Springer-Verlag. (1999).
8. Cover, T. M., Hart, P. E.: Nearest Neighbor Pattern Classification. IEEE Transaction on Information Theory, Vol. 13. (1967) 21-27.
9. Myles, J. P., Hand, D. J.: The multi-class metric problem in nearest neighbour discrimination rules. Pattern Recognition. (1990) 23(11): 1291-1297.
10. Witten, I. H., Frank, E.: Data Mining-practical Machine Learning Tools and Techniques with Java Implementation. Morgan Kaufmann, San Mateo, CA (2000)
11. Merz, C., Murphy, P., Aha, D.: UCI Repository of Machine Learning Databases. Dept of ICS, University of California, Irvine (1997). <http://www.ics.uci.edu/mlearn/MLRepository.html>
12. Zadrozny, B., Elkan, C.: Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers. In proceedings of the Eighteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, Inc. (2001) 609-616.
13. Zadrozny, B., Elkan, C.: Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2002) 694-699
14. Knorr, E. M., Ng, R. T., Zamar, H.: In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, California. (2001) 126-135
15. Domingos, P., Pazzani, Michael J.: On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. Machine Learning, Vol. 29. (1997) 103-130
16. Henrion, M.: Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling. Uncertainty in Artificial Intelligence, North Holland, Amsterdam. (1988) 317-324.

# Recurrent Boosting for Classification of Natural and Synthetic Time-Series Data

Robert D. Vincent<sup>1</sup>, Joelle Pineau<sup>1</sup>, Philip de Guzman<sup>2</sup>, and Massimo Avoli<sup>2</sup>

<sup>1</sup> School of Computer Science, McGill University, Montreal, Quebec, Canada  
`{bert, jpineau}@cs.mcgill.ca`

<sup>2</sup> Montreal Neurological Institute, McGill University, Montreal, Quebec, Canada  
`philip.deguzman@mail.mcgill.ca, massimo.avoli@mcgill.ca`

**Abstract.** Boosted ensemble classifiers have a demonstrated ability to discover regularities in large, poorly modeled datasets. In this paper we present an application of multi-hypothesis AdaBoost to detect epileptiform activity from electrophysiological recordings. While existing boosting methods do not account automatically for the sequence information that is available when analyzing time-series data, we present a recurrent extension to AdaBoost, and show that it improves classification accuracy in our application domain.

Medical treatment design has long been the exclusive domain of clinical experts. However, recently there has been a growing interest in automatically optimizing *adaptive treatment strategies* for the management of chronic diseases. The challenge is in developing sequences of treatments which adapt to a patient's characteristics and the disease's progression [1]. Additionally, there has been much recent interest in using automatic techniques to classify neurological time-series data [2]. There are tremendous opportunities in applying automated learning and discovery techniques to these classes of problems.

The optimization of an adaptive treatment strategy can be cast as a reinforcement learning problem [1]. Reinforcement learning addresses the problem of optimizing action sequences in dynamic and stochastic systems [3]. In this paradigm, the state of the system represents the patient's medical history, and the goal is to use direct experimentation with the system to learn, for each state, the optimal treatment strategy (or *policy*). Reinforcement learning unfortunately tends to require large amounts of data to reach an optimal strategy. This is impractical where data is sparse and expensive, as is often the case with human medicine. The best way to reduce data requirements is to impose strong constraints on the state representation.<sup>1</sup> Thus a significant challenge is finding a good compact state representation for a patient's medical history.

In this paper we focus on the problem of learning a compact state representation for epileptic events. Epilepsy is a brain disorder characterized by seizures (also known as *ictal* events) resulting from episodes of abnormal electrical activity in the brain. It affects about 1% of the population [4], of which at least 25% do

---

<sup>1</sup> A secondary technique is to impose strong constraints on the policy space, but this generally requires a known state representation.

not respond to anti-epileptic medication [5]. For these non-responsive patients, treatment by electrical stimulation has recently emerged as a promising alternative therapy [6]. The technology is relatively simple: a small pacemaker-like device that applies mild electrical stimulation to the nervous system is implanted in the patient. The optimization of an adaptive treatment strategy for such a device requires a compact state representation, as it is likely that limited amounts of data will be available for learning. Therefore we seek methods for classifying epileptic states from electrical field potential recordings.

In this paper we attempt to detect epileptic states by performing classification problem over fixed time frames, and we investigate the use of boosting techniques to discover information about key features for our state representation. Though this is not always well recognized, ensemble methods such as AdaBoost provide a principled and efficient mechanism for feature selection in large, poorly modeled datasets [7,8]. However, existing boosting methods do not naturally account for the sequential nature of time-series data, such as electrophysiological recordings. We present a new recurrent formulation of AdaBoost, in which the classification of prior time frames is included in the feature vector of the current time frame. This technique distinctly improves classification accuracy in our application, especially the detection of rare events. We also evaluate the performance of recurrent AdaBoost using a synthetic dataset from the UCI database [9] and demonstrate improved classification accuracy compared to standard AdaBoost. While we do not provide a formal analysis of the properties of boosting under the recurrent formulation, this will be an interesting line of future research.

## 1 Problem Description

Epileptiform signals can be separated into long *normal* phases, with periodic *ictal* events that may span several minutes. They are also characterized by brief *interictal* events, sometimes called *spikes*.

The problem of automated real-time detection and prediction of epileptic seizures using electrophysiological recordings has been investigated extensively, yielding a variety of approaches, including neural networks [10], wavelet methods [11], and nonlinear time series analysis [12]. However these results are not sufficiently interpretable to build compact state representations.

### 1.1 Data Recordings

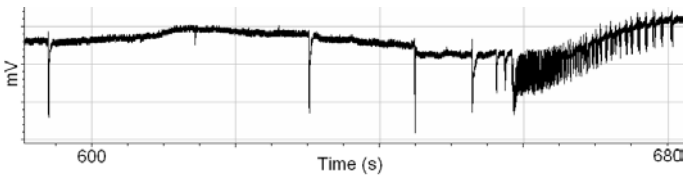
The data used in this study are field potential recordings of seizure-like activity in slices obtained from rat brains [13]. The recordings were made using micro-electrodes inserted in the regions of interest and sampled at a rate of 5012.5 Hz. The recordings were filtered to roll off frequencies above 100 Hz. This study used three separate brain slices. In each slice, neural activity was recorded in three different channels placed in different brain structures, thus yielding a total of nine data traces. These recordings are between 10.5 and 13 minutes in length.

## 1.2 Signal Processing

Each data trace was processed as a series of nonoverlapping frames consisting of 4096 samples (0.82 sec). Each frame was normalized by subtracting the mean and dividing by the full range of the entire frame. The per-frame mean, range, and energy (the sum of squared deviations from the mean) were saved for use as features in the classification. Each frame was then apodized with a Hann window and converted to a power spectrum using the discrete fast Fourier transform. Because the signals were low-pass filtered at 100Hz, only the first 80 frequency bands were used as features, representing a frequency range of approximately 1-98 Hz. The real and imaginary components of each band of the FFT were combined into a single magnitude, giving 83 features per frame (the frequency bands, plus mean, range and energy). Each trace yielded between 731 and 947 usable frames, for an overall total of 7692 frames.

## 1.3 Labeling

Each of the channels of the recordings was segmented into *normal*, *spike*, or *ictal* (or seizure) periods based on guidance from an expert. This classification was somewhat qualitative and performed by visual analysis. As can be seen in Fig. 1, the events are reasonably distinctive. Spikes were noted only for the duration of the most prominent portion of the spike waveform, giving a typical spike length of 50 milliseconds. The majority (82%) of the frames was classified as normal, with about 3% classified as an interictal spike and 14% classified as ictal. We have also made this labeled dataset publicly available [14].



**Fig. 1.** An example recording, showing several spikes and an ictal event (far right)

## 2 Algorithmic Approach

Boosting is a general supervised learning technique that seeks to combine an ensemble of simple, easily chosen classification rules (or *hypotheses*) into a single strong hypothesis. Most boosting algorithms proceed in a series of rounds in which a new weak hypothesis is trained according to a labeled set of training examples. After each round, the distribution of the training examples is updated to increase the weights of those examples that were improperly classified in the current round. The final strong hypothesis is formed by a weighted combination of the weak hypotheses [15].

## 2.1 AdaBoost

The general boosting framework specifies neither how distributions and weights are updated, nor how the weak hypotheses are to be combined. The AdaBoost (“adaptive boosting”) algorithm was invented by Freund and Schapire [7]. Our work uses AdaBoost.MH (illustrated as Algorithm 1), which is a multiclass extension of AdaBoost [16] that generalizes both the distributions and the weak learners over a set of possible labels. We specifically use “real” AdaBoost.MH, which outputs a real-valued confidence prediction for each class.

Our choice of AdaBoost was motivated primarily by the relative simplicity of the final classifier. While perhaps less amenable to human interpretation than a decision tree, a boosted classifier can yield insights into the structure of a poorly characterized problem by weighting features according to their discriminative power [8]. Also, while the algorithm’s performance is influenced by the choice of weak learners, the final strong hypothesis can often be evaluated very efficiently.

The use of the AdaBoost family of algorithms was also influenced by recent work in music genre classification which revealed AdaBoost as a powerful classification approach for complex time-series signals [17].

We use the freely available AdaBoost.MH implementation BoosTexter 2.1 [18], which includes weak learners consisting of simple decision stumps over continuous attributes. While this implementation was intended for text processing applications, it is general enough for our application.

We use the features described in Sect. 1.2 to form the feature domain  $X$ .

---

### Algorithm 1. Discrete AdaBoost.MH [16]

---

Given:  $(x_1, Y_1), \dots, (x_m, Y_m)$  where  $x_i \in \mathcal{X}$ ,  $Y_i \subseteq \mathcal{Y}$

Initialize  $D_1(i, \ell) = 1/(mk)$

**for**  $t = 1, \dots, T$  **do**

    Train weak learner using distribution  $D_t$

    Get weak hypothesis  $h_t : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$

    Choose:

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 + r_t}{1 - r_t} \right), r_t = \sum_{i, \ell} D_t(i, \ell) Y_i[\ell] h_t(i, \ell)$$

    Update:

$$D_{t+1}(i, \ell) = \frac{D_t(i, \ell) \exp(-\alpha_t Y_i[\ell] h_t(x_i, \ell))}{Z_t}$$

    (Where  $Z_t$  is a normalizing constant chosen such that  $\sum_i D_{t+1}(i) = 1$ )

**end for**

Output final hypothesis:

$$H(x, \ell) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x_i, \ell) \right)$$


---

## 2.2 Recurrent AdaBoost

AdaBoost does not directly represent any dependencies between events; each training example is assumed to be drawn independently randomly from the set  $X$ . For time series data it is likely that the classification of prior frames in the series will provide useful information for the classification of later frames.

The most obvious way to test this is to incorporate features from prior time frames  $x_{i-1}, \dots, x_{i-N}$  with features of the current time frame  $x_i$ . This method, which we call *AdaBoost with Memory*, is conceptually simple and maintains the good theoretical properties of boosting. However, it scales badly for domains with a large feature space.

Instead, we propose to use the *classification labels* of prior time frames. We train a classifier  $f$  such that  $y_i = f(x_i, y_{i-1}, \dots, y_{i-N})$ , where  $x_i$  is the input feature of frame  $i$ ,  $N$  is the number of prior predictions considered, and  $y_i$  is the set of real numbers corresponding to the class membership scores output by AdaBoost.MH. We call this algorithm *Recurrent AdaBoost*. It scales nicely with history size, assuming a small number of classes (3 in our case). A problem with  $K$  classes and  $N$  recurrent time steps adds  $NK$  features to the input vector.

Our recurrent approach requires inserting two steps in the AdaBoost training procedure. First, during initialization we set all of the prior labels in our training examples to zero. Second, these labels must be updated at the end of each round of training. The testing procedure also must be modified slightly in cases where test frames are processed in a batch manner. It is necessary to iterate classification of the test set (up to  $N$  times) to allow full incorporation of the classifier information. This is not necessary when test examples are presented in an order consistent with the time-series.

## 3 Experimental Evaluation

### 3.1 Method

In this section, we investigate the performance of boosting for the classification of epileptic brain activity from electrophysiological signals. We consider three different classification approaches:

$$\begin{array}{ll} y_i = f(x_i) & \text{Standard AdaBoost} \\ y_i = f(x_i, x_{i-1}, \dots, x_{i-N}) & \text{AdaBoost with Memory} \\ y_i = f(x_i, y_{i-1}, \dots, y_{i-N}) & \text{Recurrent AdaBoost} \end{array}$$

In the control experiment, which we call *Standard AdaBoost*, each feature vector includes the 83 scalar values associated with the current time frame only.

In the second experiment, which we call *AdaBoost with Memory*, each feature vector includes the features of both the current time window and the prior time window for a total of 166 scalar values. This method can be extended to longer memory, but we did not try this because of the substantial training time required.

In the third experiment, which we call *Recurrent AdaBoost*, the input feature vector includes the 83 standard features with the addition of the output weights for each class, for each of  $N$  prior windows (where we vary  $N$  from 1 to 5.)

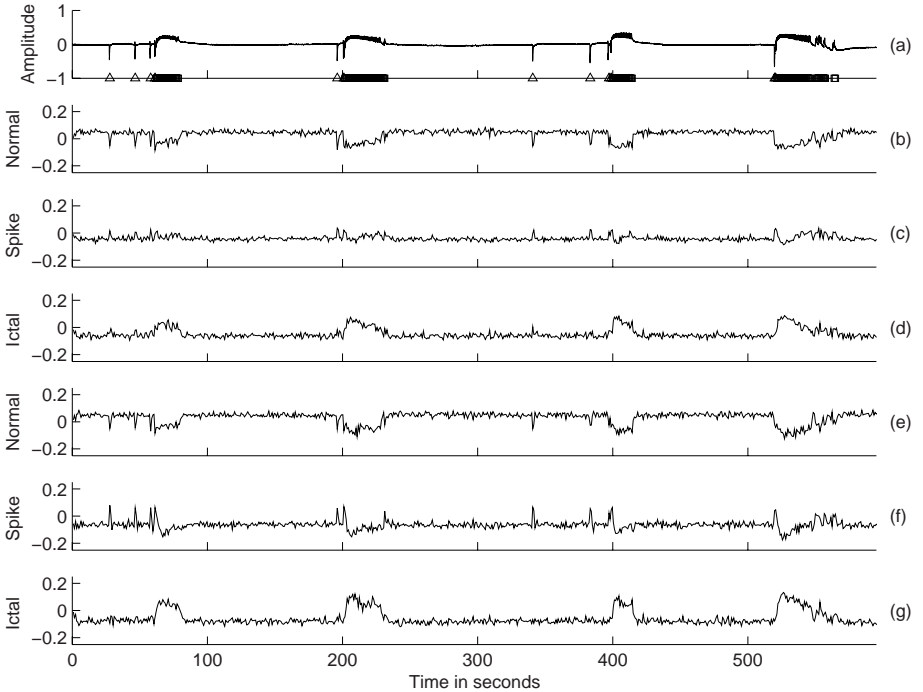
In each experiment, we performed three train/test folds using six traces as the training set and three traces as the test set. Training proceeded for 300 rounds, as the classification error leveled off after that point.

### 3.2 Results

We begin by considering an illustrative example. Figures 2b, 2c, and 2d show the classifier outputs for a representative test trace, using Standard AdaBoost. While overall results in this case were good (93% accuracy), only 10 of 12 spike frames (83%) and 82 of 119 ictal frames (69%) were correctly classified.

Figures 2e, 2f, and 2g show classifier outputs using Recurrent AdaBoost with the predictions of two prior frames. Here all 12 spike frames were properly identified, and the recognition of ictal frames increased to 102 out of 119 (86%).

We now present a more formal comparison of the approaches. We achieved average overall accuracy greater than 90% with all methods considered.

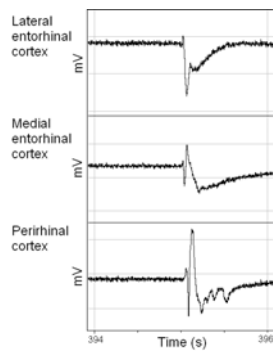


**Fig. 2.** Results for classification of one channel. (a) The original trace. (b) (c) (d) The per-class confidence values using Standard AdaBoost. (e) (f) (g) The per-class confidence values using Recurrent AdaBoost with two prior time frames.



Results for all cases are summarized in Table 1. For Standard AdaBoost, the variance in accuracy among train/test folds was relatively high, ranging from 90% to 97%. Recognition of spikes was quite poor. Spike events may be especially difficult for our detector, because of both their short duration and their relatively rarity (3% of all frames). In some cases the classifier tended to classify spikes as ictal events. This may reflect variability in the spikes, which can resemble brief ictal events (see Fig. 3).

In the AdaBoost with Memory case, all features from the prior frame are concatenated with all features from the current frame. This approach shows a large improvement over Standard AdaBoost, and markedly reduced the variance in the accuracy. Note especially the improved detection of interictal spikes.



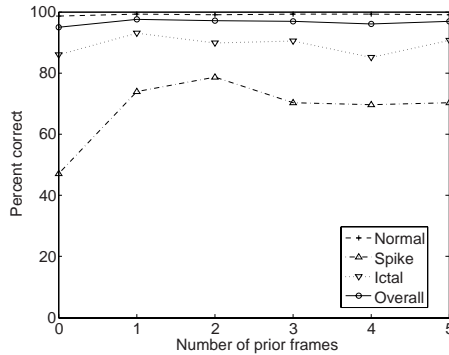
**Fig. 3.** Three channels recording a spike at approximately the same time. The bottom spike shows a long “ictal” tail.

**Table 1.** Summary of experimental results. Row labels reflect ground truth, column labels reflect classification results.

Experiment		Normal	Spike	Ictal	Total	Class%	Overall%	Range%
Standard AdaBoost	Normal	<b>6209</b>	18	67	6294	99	95	90–97
	Spike	35	<b>119</b>	99	253	47		
	Ictal	97	65	<b>983</b>	1145	86		
AdaBoost with Memory	Normal	<b>6242</b>	15	37	6294	99	97	93–99
	Spike	48	<b>187</b>	18	253	74		
	Ictal	92	15	<b>1038</b>	1145	91		
Recurrent AdaBoost (1 prior)	Normal	<b>6253</b>	16	25	6294	99	98	94–99
	Spike	49	<b>187</b>	17	253	74		
	Ictal	69	12	<b>1064</b>	1145	93		
Recurrent AdaBoost (2 prior)	Normal	<b>6239</b>	22	33	6294	99	97	92–99
	Spike	42	<b>199</b>	12	253	79		
	Ictal	101	15	<b>1029</b>	1145	90		
HMM	Normal					97	94	
	Spike					45		
	Ictal					78		

Results for Recurrent AdaBoost are shown for two cases, incorporating the predictions for either one or two prior frames. Incorporating one prior frame, there is a strong improvement over Standard AdaBoost in classifying both spikes and ictal events. Incorporating two prior frames provides no consistent benefit. These results are comparable to those of AdaBoost with Memory, but with less training time, given the smaller size of the feature space.

We evaluated Recurrent AdaBoost when incorporating predictions for 1–5 prior frames into the feature vector. These results are summarized in Fig. 4. There is little improvement beyond two frames, suggesting that, for our dataset, there is little added information in more distant time frames.



**Fig. 4.** Results for Recurrent AdaBoost using varying numbers of prior frames

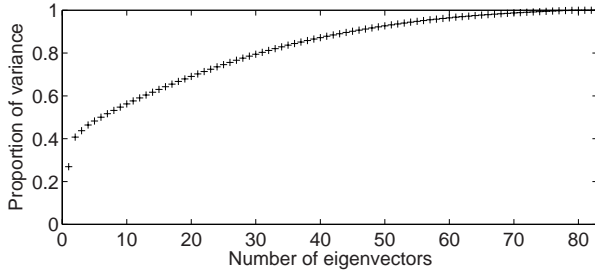
These results show solid detection of the various epileptic states. However, spike detection accuracy is at most 79%. In conversations with experts with years of experience reading recordings of the type shown in Fig. 1, the consensus is that given a full trace it is fairly easy to differentiate spiking and ictal events. However, given only data up to time  $t$ , it is difficult to predict whether a subsequent burst of energy is either a spike or the onset of an ictal event.

We also show results obtained using a standard time-series approach, the Hidden Markov Model (HMM) [19].<sup>2</sup> The results for classification of spike events are comparable to those of Standard AdaBoost but significantly worse than those of Recurrent AdaBoost. Classification of ictal events is worse than both Standard and Recurrent AdaBoost.

### 3.3 Feature Extraction

To better characterize the data, we performed principal components analysis of the 83 features that form our input space. Figure 5 shows that at least 50% of the principal components are required to reconstruct 90% of the variance

<sup>2</sup> Observation probabilities were modeled assuming each input feature follows a univariate Gaussian. All parameters were derived from the labeled training data.



**Fig. 5.** Principal components analysis of 83 features

in our data. We also observed much overlap between the spike class and the ictal/normal classes in principal components space. Both this analysis and the prior literature on the topic suggest that epileptic state detection from electrical signals is difficult, especially for spike events.

We also examined the strong hypotheses produced by AdaBoost.MH for all of the experiments. We observed a number of striking regularities.

In all recurrent examples, the first weak hypothesis recruited was either frequency band 62 or 63, corresponding to frequencies of 76 or 77 Hz. High values in these bands favor a normal classification, whereas low values weight towards ictal classification. Frequency bands 6–8 ( $\sim 7$ –10 Hz) were consistently recruited early. Low values in these bands favor normal classification, whereas high values favor ictal classification.

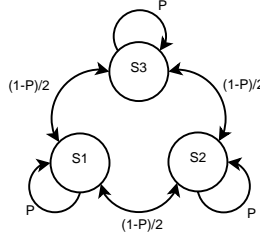
In most cases, energy was recruited in the first 20 rounds. A high energy value resulted in a strong weighting toward a spike classification. A similar effect was seen for the range feature.

In recurrent cases, prior labels primarily acted as a source of hysteresis in the system: prior labels of ictal or normal biased the present frame towards either ictal or normal, respectively.

### 3.4 Validation

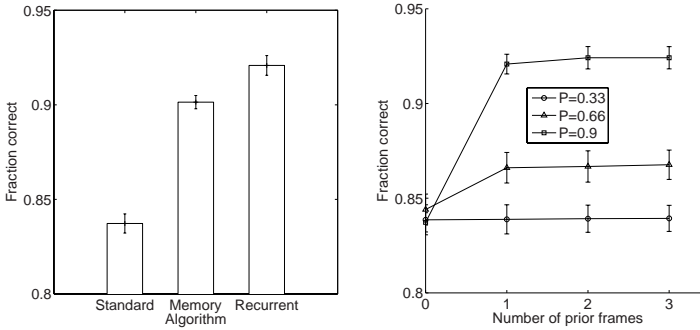
We also performed validation of Recurrent AdaBoost with synthetic data by adapting a similar task of known difficulty, the “waveform” classification problem from the UCI Machine Learning repository [20,9,21]. This task requires discrimination among three classes of 21 noisy continuous features. When the classes are chosen uniformly randomly in sequence (i.e. when the examples are generated i.i.d.), Bayes optimal performance is 86% [9]. We modified the problem by sampling examples from these three classes in a nonuniform sequence using the simple Markov model illustrated in Fig. 6 so as to produce time-series data.

We repeated each of the algorithms using 12 train/test rounds of 5000 training examples and 1000 testing examples, varying the transition probability  $P$ .



**Fig. 6.** Simple Markov model used to generate synthetic data. The parameter  $P$  sets the probability that the output class of an example is the same as that of the prior example. Initial states are chosen uniformly randomly.

The results are summarized in Fig. 7. The bar graph on the left shows the results for  $P = 0.9$  for Standard AdaBoost (left), AdaBoost with Memory (middle), and Recurrent AdaBoost using one prior prediction (right). The line graph on the right shows the results for Recurrent AdaBoost when  $P = 0.33$ ,  $P = 0.66$ , and  $P = 0.9$ , varying the number of prior predictions.



**Fig. 7.** Results for experiments with synthetic data. Error bars show 95% confidence intervals over 12 folds.

When no prior information is used, the results are similar to the Bayes optimal value (86%) for the uniformly random case; this is the Standard AdaBoost algorithm. However, when prior information is incorporated, we achieve significantly improved performance, as the algorithm is able to exploit the time-series information we added to the problem. Unsurprisingly, the improvement is largest when the value of  $P$  is highest, and only a single prior prediction is needed to capture the first-order Markov model. Recurrent AdaBoost performs slightly better than AdaBoost with Memory on this domain. We speculate that Recurrent AdaBoost achieves this by forming a smoothed summary of the state history.

## 4 Discussion

We propose a new way to apply boosting to time-series data by recurrent incorporation of class predictions into the feature vector. We show that this approach improves classification results in experiments with both real and synthetic data. We also contribute a new labeled dataset for time-series classification [14].

We also provide the first empirical evidence that AdaBoost can be used to characterize epileptic states in neurophysiological recordings. This task is difficult because of the large feature space, the unbalanced class distributions, the limited availability of training data, and the great variability of these recordings.

These findings show robust detection of key epileptic states. Recognition of interictal spikes was the most problematic, exhibiting high variance over the test cases. Note however that the training set is very small for this class, at most 204 examples for an 83-dimensional feature space. Furthermore, the class has strong overlap with others over the principal components of the feature space. In the future, we hope to investigate whether a similar approach may be used to classify subtler signals, such as those of cognitive states.

Our investigation was limited to using very simple weak learners. There is evidence that more sophisticated weak learners may yield a better strong hypothesis [17]. Other methods for applying boosting to time series data involved modifying the weak learners to account for time or spatial relationships [22,23]. This may be something to consider in the future.

We do not at this time provide a formal analysis of the convergence properties of Recurrent AdaBoost. The main challenge is the fact that the input set is not stationary due to its dependence on the classification of prior instances. This raises interesting theoretical questions which will be addressed in the future.

**Acknowledgments.** The authors gratefully acknowledge the financial support of the Canadian Institutes of Health Research (CIHR) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

## References

1. Murphy, S.A.: An experimental design for the development of adaptive treatment strategies. *Statistics in Medicine* **24** (2005) 1455–1481
2. Mitchell, T.M., Hutchinson, R., Niculescu, R.S., Pereira, F., Wang, X., Just, M., Newman, S.: Learning to decode cognitive states from brain images. *Machine Learning* **57** (2004) 145–175
3. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* **4** (1996) 237–285
4. Hauser, W.A., Hesdorffer, D.C.: *Epilepsy: Frequency, Causes and Consequences*. Demos, New York (1990)
5. Kwan, P., Brodie, M.J.: Early identification of refractory epilepsy. *New England Journal of Medicine* **342**(5) (2000) 314–319
6. Uthman, B.M., Reichl, A.M., Dean, J.C., Eisenschenk, S., Gilmore, R., Reid, S., Roper, S.N., Wilder, B.J.: Effectiveness of vagus nerve stimulation in epilepsy patients: a 12 year observation. *Neurology* **63** (2004) 1124–1126

7. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55** (1997) 119–139
8. Viola, P., Jones, M.: Robust real-time object detection. *International Journal of Computer Vision* **57**(2) (2004)
9. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. *Machine Learning* **6** (1991) 37–66
10. Chiu, A.W.L., Daniel, S., Khosravani, H., Carlen, P.L., Bardakjian, B.L.: Prediction of seizure onset in an *in-vitro* hippocampal slice model of epilepsy using gaussian-based and wavelet-based artificial neural networks. *Annals of Biomedical Engineering* **33**(6) (2005) 798–810
11. Khan, Y.U., Gotman, J.: Wavelet based automatic seizure detection in intracerebral electroencephalogram. *Clinical Neurophysiology* **114** (2003) 898–908
12. Martinerie, J., Adam, C., Le Van Quyen, M., Baulac, M., Clemenceau, S., Renault, B., Varela, F.J.: Epileptic seizures can be anticipated by non-linear analysis. *Nature Medicine* **4**(10) (1998) 1173–1176
13. De Guzman, P., D’Antuono, M., Avoli, M.: Initiation of electrographic seizures by neuronal networks in entorhinal and perirhinal cortices *in vitro*. *Neuroscience* **123** (2004) 875–886
14. Vincent, R.D., Pineau, J., de Guzman, P., Avoli, M.: Labeled epileptiform data. <http://www.cs.mcgill.ca/~jpineau/datasets/epilepsy.tar.gz> (2006)
15. Schapire, R.E.: The boosting approach to machine learning: an overview. In Denison, D.D., Hansen, M.H., Holmes, C.C., Mallick, B., Yu, B., eds.: *Nonlinear Estimation and Classification*. Springer (2003) 149–172
16. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. *Machine Learning* **37** (1999) 297–336
17. Bergstra, J., Casagrande, N., Eck, D.: Two algorithms for timbre- and rhythm-based multi-resolution audio classification. In: *1st Annual Music Information Retrieval Exchange*. (2005)
18. Schapire, R.E., Singer, Y.: BoosTexter: A boosting-based system for text categorization. *Machine Learning* **39**(2/3) (2000) 135–168
19. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**(2) (1989)
20. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA (1984)
21. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html> (1998)
22. Diez, J.J.R., González, C.A.: Applying boosting to similarity literals for time series classification. In Kittler, J., Roli, F., eds.: *Multiple Classifier Systems*. Volume 1857 of *Lecture Notes in Computer Science*, Springer (2000) 210–219
23. Boné, R., Assaad, M., Crucianu, M.: Boosting recurrent neural networks for time series prediction. In Pearson, D., Steele, N.C., Albrecht, R.F., eds.: *Artificial Neural Networks and Genetic Algorithms*. Springer (2003) 18–22

# Pattern Classification in No-Limit Poker: A Head-Start Evolutionary Approach

Brien Beattie, Garrett Nicolai, David Gerhard, and Robert J. Hilderman

University of Regina  
Department of Computer Science  
Regina, SK Canada S4S 0A2  
{beattieb,nicolaig,gerhard,hilder}@cs.uregina.ca

**Abstract.** We have constructed a poker classification system which makes informed betting decisions based upon three defining features extracted while playing poker: hand value, risk, and aggressiveness. The system is implemented as a player-agent, therefore the goals of the classifier are not only to correctly determine whether each hand should be folded, called, or raised, but to win as many chips as possible from the other players. The decision space is found by evolutionary methods, starting from a data-driven initial state. Our results showed that evolving an agent from a data-driven “head-start” position resulted in the best performance over agents evolved from scratch, data-driven agents, random agents, and “always fold” agents.

**Keywords:** Evolution, Pattern Classification, No-limit Hold'em, Poker.

## 1 Introduction

In recent years, poker has become a growing phenomenon. Within the world of poker itself, the game of Texas hold'em, and in particular, no-limit Texas hold'em, has arisen as perhaps the favourite among amateur players.

Texas hold'em is slightly different from other variations of poker. It has two significant properties which contribute to its popularity. First, the simplicity of the game means that memorization techniques are not nearly as important as in such variants as stud poker. Second, the use of “community” table cards which are available to all players ensures a measure of similarity between the qualities of the hands. This automatic balancing of the game makes tactics and strategy a much more important component of the game.

Poker lends itself well to classification, but due to the peculiarities noted above, Texas hold'em presents a unique challenge to the classification problem. If a player were only playing to win hands, then a straight classifier would be useful for determining the player's actions. However, when a player is trying to win as much money as possible, it no longer makes sense to raise every time that the player has very good cards as a straight classifier undoubtedly would. If a player will likely win regardless of the cards to come, the hand might be better

played by playing the cards as though they were merely mediocre in an attempt to draw more money out of the opponents.

Another problem can arise when a player always makes the same moves in the same situations. The player becomes predictable. Predictable players can be exploited, and as such, can be defeated by intelligent players regardless of their hands. The classification problem lies not in the ability to predict which action is the most beneficial in a given situation, but rather in the ability to win money for the player over the long term, possibly gaining less money in the short.

### 1.1 Recent Work in the Area

Several attempts have been made to make a successful poker-playing agent using various strategies to take factors such as playing styles, bluffing, and predictability into account. Billings *et al.* [1,2] created the *Poki* system. Although *Poki* plays limit hold'em, and our program plays no-limit, several similarities are present in our architectures. *Poki* uses opponent modeling and hand evaluation to arrive at betting decisions. It is also capable of bluffing in particular situations. Unlike our system, much of the decision making in *Poki* is rule-based, and thus the best choice will be made every time, making the player somewhat predictable. This practice is acceptable for limit hold'em since monetary risk is limited. Unfortunately, it becomes unviable when no-limit hold'em and unlimited bets are considered.

The agent developed by Blank *et al.* [3] was also for limit hold'em. They use support vector machines to create three classifiers: fold vs. call, fold vs. raise, and call vs. raise. A voting system is used to choose the most advantageous betting choice. In the case of ties, confidence of the decisions is used as a tie-breaker. Unlike our system, predictability is not considered, and the system presented merely selects the best solution from the candidates. Again, this is acceptable for limit hold'em, but could quickly prove dangerous in no-limit.

Oliehoek [4] uses co-evolution in an attempt to minimize the worst-case payout between two poker players. Unlike our system, Oliehoek concentrates on a simplified variation of poker, using a deck of 8 cards and a single betting round.

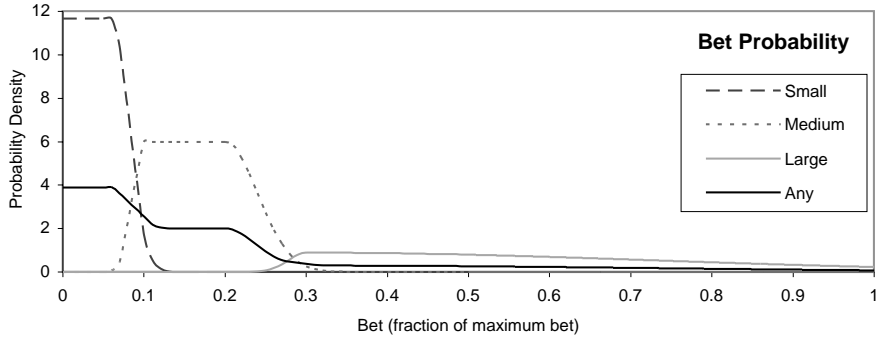
Southey *et al.* [5] use several different methods, including Bayes classification and *a-priori* classification to determine the best selection in a 2-player limit tournament. It seems that limit poker is a variant that is investigated much more often than no-limit.

Although limit and no-limit hold'em share many characteristics, play varies considerably due to the differences in betting structures. It can be said that a player is "punished" more for making a bad decision in no-limit hold'em. The cost even to see the first of the community cards can be exceedingly high. Consequently, the cost of playing incorrectly in such a situation can be extreme.

## 2 System Architecture and Design Choices

While most existing poker classifiers are built to play limit hold'em, we wanted to create a player-agent for the much more popular no-limit variant. Existing





**Fig. 1.** Raise type distributions. An agent will randomly choose a raise value based on the distribution of the selected raise type.

systems are also often designed only for the two-player or “heads up” situation. Although this is arguably the most exciting part of a hold’em game, it is only a subset of the larger game. We strove to create a player-agent that could play well against anywhere from 1 to 9 opponents, as in a true hold’em game.

Our system considers three main features: hand value, risk and aggressiveness, described below. From these features, a decision is made whether to fold, call, or raise. It does not simply perform the optimal action in each circumstance, but rather determines a probability of choosing each action: fold, call, and raise. This method of action selection allows our agent to make informed decisions, without becoming predictable to other players. Since our agent is playing no-limit hold’em and raise amounts are not pre-defined, we divide raises into three categories—small, medium, and large raises—which are considered as separate actions. The bet distributions for each of the raise categories can be seen in Figure 1.

## 2.1 Hand Value

The hand value is calculated as the probability that the current hand will beat all opponents’ hands assuming that their cards are distributed uniformly. The value is normalized so that it is independent of the number of opponents being faced. Although this calculation could prove temporally expensive, an exhaustive lookup table was created allowing the calculation to be done in constant time.

## 2.2 Risk

The risk function for our system went through several incarnations before arriving at its final state. Initially, we determined the risk as a direct ratio between the bet size and the pot size. This measure, the so-called “implied odds” used in current poker research, fails to take into account the magnitude of the bets. In cases where the bet amounts are small, as is the case immediately after blinds are posted, the risk thus calculated is evaluated as high, despite the minimal cost of calling. To address this failing, our second risk function (Equation 1) multiplied

the implied odds by the Euclidean distance from the origin normalised by the blind. This function was still inadequate because it did not consider the size of the player’s chip stack, and the range of the function was inconsistent.

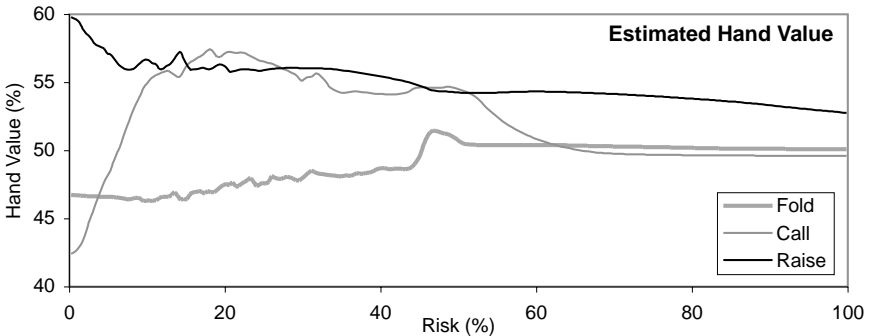
$$risk = \frac{bet\sqrt{bet^2 + pot^2}}{pot \times blind} \quad (1)$$

The final risk function (Equation 2) replaced the pot size with the future pot size, calculated by adding the player’s bet, should he call. The Euclidian distance previously used to account for the magnitude of the bet was changed to a Manhattan distance. The normalizing factor—the maximum pot, determined as the amount of the pot should the player go all-in and all remaining players call—was changed to take players’ chip counts into consideration. The result was then multiplied by  $\frac{4}{3}$  to set the range of the values between zero and one. The final risk amounts to a multiplicative average, hence the presence of the square root in the formula.

$$risk = \sqrt{\frac{4}{3} \times \frac{bet(2bet + pot)}{maxpot(bet + pot)}} \quad (2)$$

### 2.3 Aggressiveness

Most other systems observed have some sort of player-modeling which we represent in our aggressiveness feature. Whenever an opponent makes a decision (fold, call, or raise), our agent determines the opponent’s most probable hand value. We do this by assuming that opponents play in the same way as our agent. The most probable hand value can be determined as the average hand value for which our agent would make the same decision while facing the same risk. Figure 2 shows how one agent estimates these probable hand values. Opponents’ general aggressiveness can be modeled as the deviation of their estimated hand values from their expected hand values, assuming their hands are uniformly distributed.



**Fig. 2.** Opponent Hand Estimations for aggressiveness based on perceived risk

The aggressiveness of a particular action is evaluated as the most probable hand value for that action normalized by the player's general aggressiveness, as seen in Equation 3, where  $A$  is the player's calculated aggressiveness,  $L$  is the aggressiveness of the player's last action, and  $G$  is the player's general aggressiveness. Our system does not use opponent aggressiveness directly as a factor in its decision making. Instead, the player's hand value is modified by the average aggressiveness of the table in the current hand according to in Equation 4, where  $V_2$  is the modified hand value,  $w$  is the aggressiveness weight, and  $V_1$  is the original hand value.

$$A = L^{\frac{-1}{\log_2(G)}} \quad (3)$$

$$V_2 = w \times V_1^{\frac{-1}{\log_2(A)}} + (1 - w) \times V_1 \quad (4)$$

### 3 Training and Testing Data

We worked under the hypothesis that evolutionary methods would produce the best results for training our classifier. The principal benefit of evolutionary methods is that enormous data sets are not needed for training. The evolutionary process essentially generates its own training data. Agent evolution can be given a head start by training a data-driven agent from a small, hand-derived set of training data, and applying evolutionary techniques from that data-driven agent.

To create the data-driven agent (the head-start for evolution), a source of training data was needed. All public database recordings of poker games that we found were presented from a spectator's point of view. Unfortunately, the spectator does not get to see players' cards in many, if not most, circumstances. This missing information is critical for training a system properly. As an alternative, we chose to record data from televised poker games, where all players' cards are shown at all times. In this manner, an adequate data set was collected for training our data-driven agent.

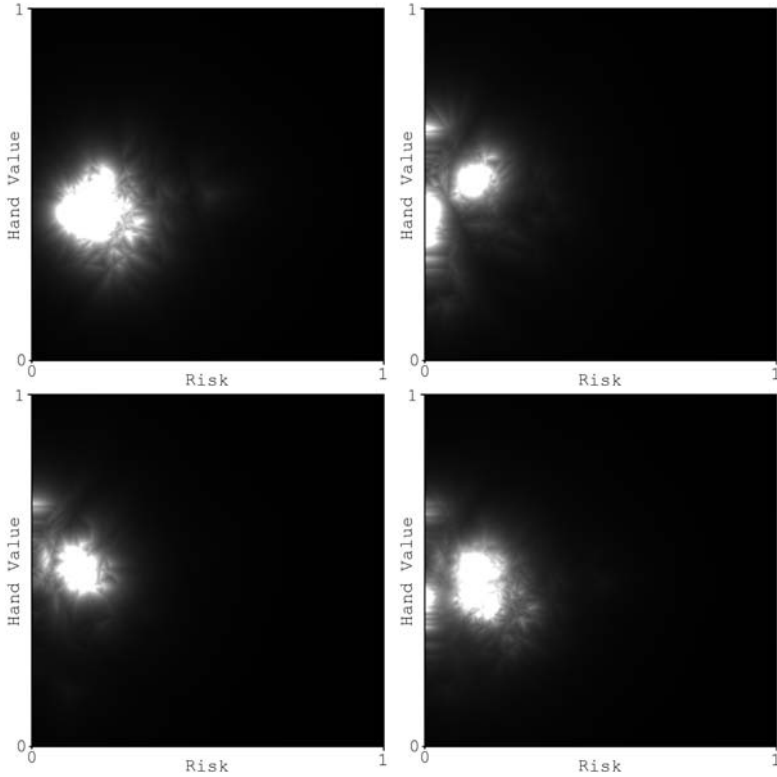
### 4 Implementation

A  $k$ -nearest neighbour estimation [6] was used to approximate the densities of folds, calls, and raises. These density estimations are presented in Figure 3.<sup>1</sup>

A classifier can be constructed from the density estimations of the various actions. The density of each action is divided by the total density (obtained by summing the densities of each of the actions) to obtain a relative probability for choosing that action. These relative probabilities define our agent's probable action in any given situation (defined by its risk and hand value). The relative probabilities obtained by this method are presented in Figure 4.

These action topographies were decided upon as the best way of defining our classifier because they most effectively cover the problem domain. Neural

<sup>1</sup> For high-quality colour versions of these and other topographies appearing in this paper, see <http://www2.cs.uregina.ca/~gerhard/research/poker.html>

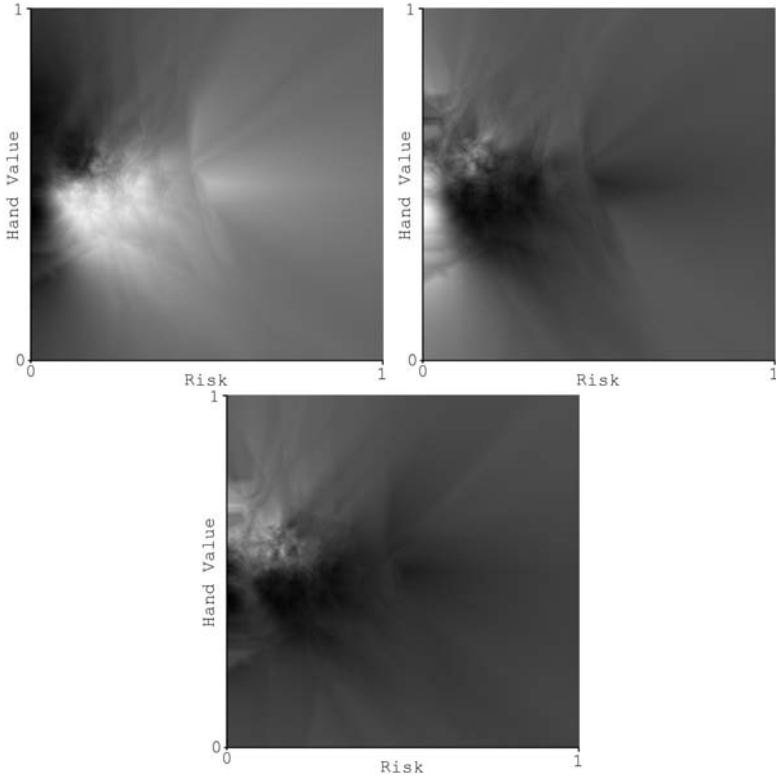


**Fig. 3.** Action distribution density estimations. (top-left) folds; (top-right) calls; (bottom-left) raises; (bottom-right) combined distribution of all actions. Densities range from zero (black) to maximal density (white).

networks require considerable training data, and with our limited data set, we expected that the networks would be incorrectly trained leading to poor poker agents. Decision trees would unnecessarily condense our data. Furthermore, with only three feature values, a decision tree would not adequately separate our data without being overly large.

We set up an evolutionary process wherein each generation involves a set of agents playing a number of tournaments. For each generation, the agents with the best results from the previous generation are carried forward and used as parents to generate the remaining agents for that generation. For the first generation, the data-driven agent is used as the sole evolutionary parent for the remaining agents.

New agents created from parents carried forward from previous generations are generated by common evolutionary methods, both recombination and mutation. Combination of parents is achieved by using a weighted average of their action topographies. It should be noted that a child need not have two parents but could have any number including one.



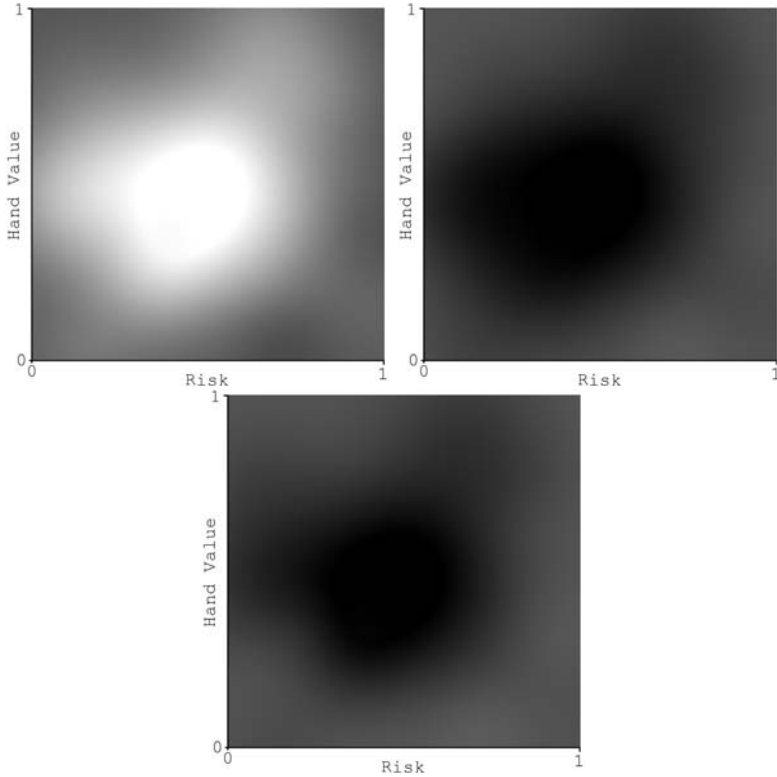
**Fig. 4.** Action topographies of the data-driven agent. Clockwise from top left: folds, calls, and raises.

Mutation was a bit more difficult to envision and implement. Unlike other evolutionary algorithms which simply flip a bit to simulate mutation, we did not want to change values across the probability tables. Such high frequency noise would result in inconsistency in an agent’s playing habits. To counter this form of sporadic play, we developed a “hill function” which would generate a number of random positive or negative hills. Such hill functions provide a more desirable low frequency noise which results in more meaningful mutations.

It should be noted that the agents that survived from the last generation are not mutated at all so as to preserve the best agents from the previous round of tournaments. It may occur that none of the children are better than the parents, in which case the parents should be kept.

## 5 Comparison and Results

We created several different types of agents in addition to our initial data-driven agent, although for testing purposes, we consider three of interest. The first is



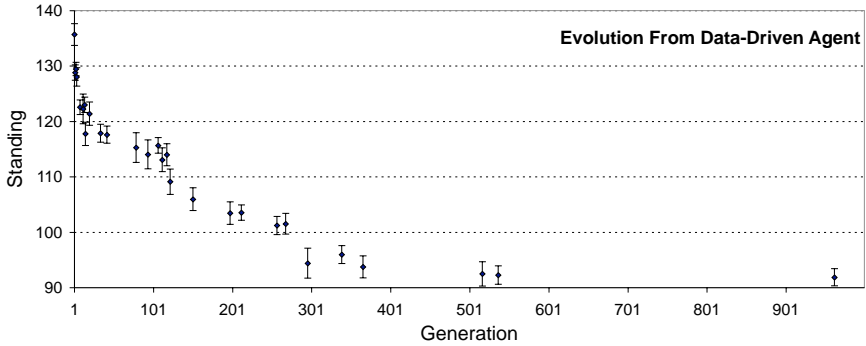
**Fig. 5.** Action topographies evolved from uniform topographies. Clockwise from top left: folds, calls, and raises.

the data-driven agent described earlier which was generated from the collected test data. The second was generated using our evolutionary algorithm starting with uniform topographies, and the third was generated using our evolutionary algorithm with the data-driven agent as a head-start.

The agent evolved from scratch would possess no inherent biases which may have been present in the data collected. The result was a set of action topographies which appear quite smooth when compared to others. These topographies can be seen in Figure 5.

It can be seen that although the evolution started out with uniform topographies, structured topographies developed and a competent player-agent emerged. The agent, as it turns out, is not only better than random but better than the data-driven agent already developed.

A one-thousand generation evolution was run starting with the data-driven agent. The process showed a marked improvement in the leading agents which slowed down as the generations passed. Over the course of the one thousand generations of evolution, twenty seven unique agents emerged. Afterward, these



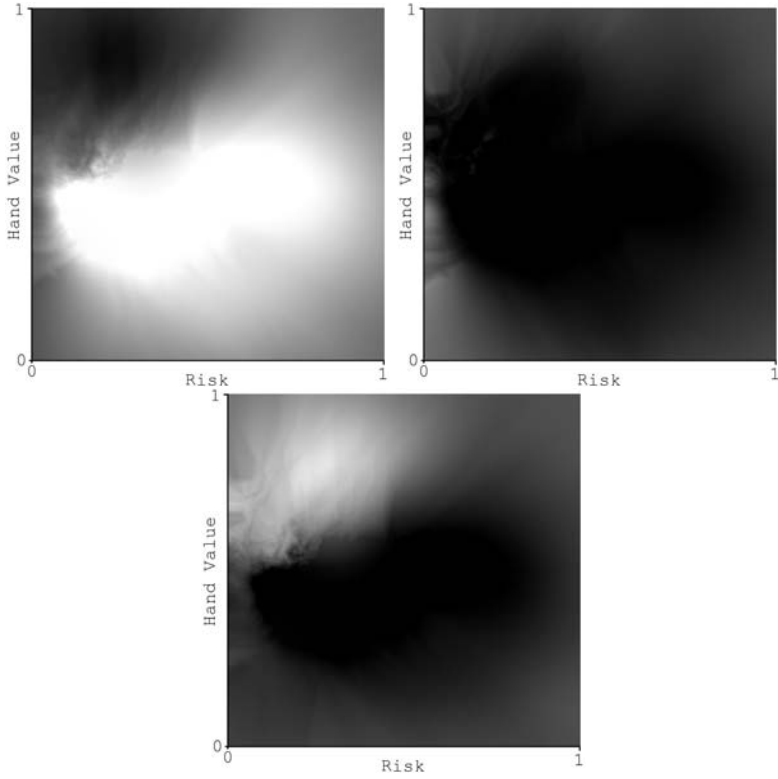
**Fig. 6.** Tournament results of agents evolved from the data-driven agent

agents, together with the data-driven agent, were duplicated eight times and played against one another in a number of tournaments (224 agents each).

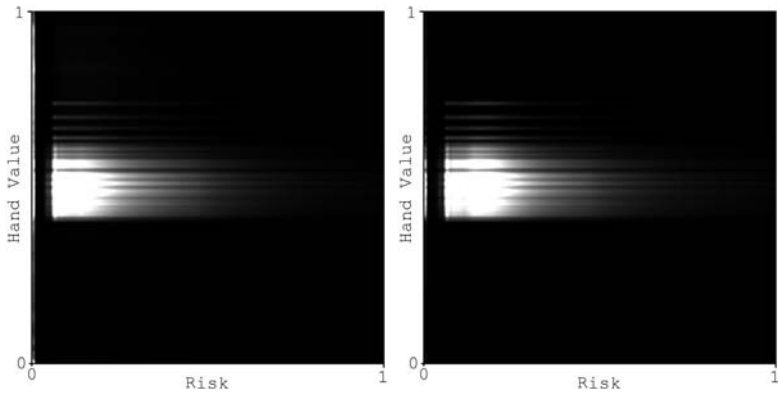
Figure 6 plots their average standing (lower is better) against the generation in which they appeared, thus demonstrating the effectiveness of the evolution. The final action topographies for our evolved agents can be seen in Figure 7. It can be seen that folds tend to dominate the landscape. Although this result might seem unrealistic, it is actually a reasonable strategy. Although online poker tends to see a lot of calls and raises, professional poker tournaments and even real-money online games often see the majority of hands folded as players wait for better cards. Calls are typically made in low risk situations. Raises, it is quite apparent, appear with high hand values. These properties are what one would expect from an intelligent player.

The aggressiveness function described earlier determines opponent aggressiveness with respect to how the agent would play a particular hand. Necessary for this calculation is the prior-probability distribution of hand value and risk. In the case of the data-driven agent, this distribution is known from the test data. For evolved agents, records from the previous generation's tournaments are used to determine the prior probabilities. These updated prior probabilities prepare the evolved agents for the situations which they actually face and which, assuming that successive generations are getting progressively more intelligent, should better reflect the real world. The prior-probability distributions used for the evolved agents of interest can be seen in Figure 8. The prior-probability distribution for the data-driven agent is in the bottom right frame of Figure 3.

To compare the performance of two agents, a set of tournaments are run, each with an equal number of agents of the two playing styles. The results of these tournaments are used to evaluate which of the two playing styles is superior. The score function seen in Equation 5 is a metric which we developed to quantify the comparison results.  $\bar{x}_1$  and  $\bar{x}_2$  are the mean standing for agents of two playing styles, and  $n$  is the total number of agents in the tournament. The function is a relative score to be considered in conjunction with the opponent against whom it was obtained. A score of 100% against random agents would imply that all of the random agents finished in the bottom half of the rankings in every tournament.



**Fig. 7.** Action topographies evolved from the data-driven agent in Figure 4. Clockwise from top left: folds, calls, and raises.



**Fig. 8.** Prior probabilities used in aggressiveness functions for evolved agents. (left) evolved from random; (right) evolved from the data-driven agent. Prior probabilities for the data-driven agent are equivalent to the combined distributions from Figure 3.



**Table 1.** Mean player-relative scores

	Random	Data-driven	Folding	Scratch
Head-start	72.5% $\pm$ 0.36%	56.1% $\pm$ 0.39%	28.4% $\pm$ 0.43%	36.6% $\pm$ 0.43%
Scratch	65.9% $\pm$ 0.36%	43.3% $\pm$ 0.42%	27.0% $\pm$ 0.45%	
Folding	79.2% $\pm$ 0.32%	65.9% $\pm$ 0.40%		
Data-driven	29.7% $\pm$ 0.44%			

A zero score implies that the opponents in that tournament were equally skilled because they each have the same mean ranking. A negative score suggests that the opponent is better than the agent being considered.

$$Score = \frac{\bar{x}_2 - \bar{x}_1}{n/2} \quad (5)$$

The three agents of interest (data-driven, evolved from random, and evolved from head-start) were all played against one another as well as two control agents: a random agent ( $\frac{1}{3}$  fold,  $\frac{1}{3}$  call,  $\frac{1}{9}$  small raise,  $\frac{1}{9}$  medium raise,  $\frac{1}{9}$  large raise) and an agent that folds every hand. The relative scores are presented in Table 1. “Head-start” is the agent evolved from the data-driven agent, “Scratch” is the agent evolved from uniform topologies. “Folding” is an agent that folds every hand, “Data-driven” is the agent trained from our collected data set, and “Random” is the random agent described above. The head-start evolved agent performed better than all the others.

## 6 Conclusions and Future Work

Although we did evolve several poker playing agents that performed well against random play and against one another, it is difficult to gauge whether or not they are good poker players. The player-agents were able to win simulated tournaments, but these tournaments were against other agents. Although the agents played capably against us, we are by no means expert poker players. The results are encouraging. Competent poker playing agents can be developed using evolutionary techniques, with playing ability improving from generation to generation.

Significant optimization of the evolutionary parameters was not done, and the parameters used may not have been the best ones. Although we did tweak the parameters slightly, we arrived at no definite conclusion regarding the best values for any given parameter. It would also be possible to take the evolution to later epochs to determine if agents can be further improved.

Considerable testing against human players is required to gain a full understanding of the capabilities and limitations of our system. Data from such testing would likely also be useful for further enhancement of the system.

Finally, an adaptation could be made allowing the system to build profiles of human players. A human player could play the game while the system builds a

strategy from his playing style. This profile could be compared to player agents, or possibly used as another head-start point for experimental evolution.

## References

1. Billings, D., Davidson, A., Schaeffer, J., Szafron, D.: The challenge of poker. *Artificial Intelligence* **134**(1–2) (2002) 201–240
2. Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., Szafron, D.: Approximating game theoretic optimal strategies for full-scale poker. In: *International Joint Conference on Artificial Intelligence*. (August 2003) 661–668
3. Blank, T., Soh, L.K., Scott, S.: Creating an svm to play strong poker. In: *International Conference on Machine Learning and Applications*. (December 2004) 150–155
4. Oliehoek, F.A., Vlassis, N., de Jong, E.D.: Coevolutionary nash in poker games. In: *17th Belgian-Dutch Conference on Artificial Intelligence*. (October 2005) 188–193
5. Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., Billings, D., Rayner, C.: Bayes' bluff: Opponent modelling in poker. In: *Twenty-First Conference on Uncertainty in Artificial Intelligence*. (July 2005) 550–558
6. Duda, R.O., Hart, P.E., Stork, D.G.: *Pattern Classification*. Wiley Interscience (2000)

# Managing Conditional and Composite CSPs

Malek Mouhoub and Amrudee Sukpan

University of Regina  
Wascana Parkway, Regina, SK, Canada, S4S 0A2  
{mouhoubm, sukpanla}@cs.uregina.ca

**Abstract.** *Conditional CSPs* and *Composite CSPs* have been known in the CSP discipline for fifteen years, especially in scheduling, planning, diagnosis and configuration domains. Basically a *conditional constraint* restricts the participation of a variable in a feasible scenario while a composite variable allows us to express a disjunction of variables or sub CSPs where only one will be added to the problem to solve. In this paper we combine the features of *Conditional CSPs* and *Composite CSPs* in a unique framework that we call *Conditional and Composite CSPs (CCCSPs)*. Our framework allows the representation of dynamic constraint problems where all the information corresponding to any possible change are available a priori. Indeed these latter information are added to the problem to solve in a dynamic manner, during the resolution process, via conditional (or activity) constraints and composite variables. A composite variable is a variable whose possible values are CSP variables. In other words this allows us to represent disjunctive variables where only one will be added to the problem to solve. An activity constraint activates a non active variable (this latter variable will be added to the problem to solve) if a given condition holds on some other active variables. In order to solve the CCCSP, we propose two methods that are respectively based on constraint propagation and Stochastic Local Search (SLS). The experimental study, we conducted on randomly generated CCCSPs demonstrates the efficiency of a variant of the MAC strategy (that we call MAC+) over the other constraint propagation techniques. We will also show that MAC+ outperforms the SLS method MCRW for highly consistent CCCSPs. MCRW is however the procedure of choice for under constrained and middle constrained problems and also for highly constrained problems if we trade search time for the quality of the solution returned (number of solved constraints).

**Keywords:** Constraint Satisfaction, Local Search, Arc Consistency.

## 1 Introduction

*Conditional CSPs* and *Composite CSPs* have been known in the CSP discipline for fifteen years, especially in configuration domains [1,2,3,4,5,6,7,8,9,10]. In fact these two frameworks are quite typical for any combinatorial problem such as planning, diagnosis, and temporal reasoning. Basically a *conditional constraint* restricts the participation of a variable in a feasible scenario while a composite variable allows us to express a disjunction of variables or sub CSPs where only one will be added to the problem to solve. In the original *Conditional CSP* framework [1], *activity constraints* are introduced to

handle conditional variables. A variable has either *active* or *non-active* status. Only active variables are required value assignments. Given two variables  $X_i$  and  $X_j$ , four types of activity constraints have been defined as follows:

1. **require var. active** :  $(X_i) \wedge ((X_i = a_{i1}) \vee \dots (X_i = a_{ip})) \rightarrow X_j$
2. **always require. active** :  $(X_i) \rightarrow X_j$
3. **require not. active** :  $(X_i) \wedge ((X_i = a_{i1}) \vee \dots (X_i = a_{ip})) \xrightarrow{rn} X_j$
4. **always require not. active** :  $(X_i) \xrightarrow{rn} X_j$

The activity constraint (1) will activate  $X_j$  if the active variable  $X_i$  is assigned one of the values  $a_{i1} \dots a_{ip}$  from its domain whereas the activity constraint (2) will activate  $X_j$  if  $X_i$  is active. In (3) and (4), if the condition is true,  $X_j$  is not required in a solution, or is set to a *non-active* variable. The following three systematic resolution techniques have been proposed for *Conditional CSPs*.

1. A CCSP is converted into a CSP by reformulating an activity constraint by a compatibility constraint (by adding a *NULL* value to the domain of the conditional variables). Classical CSP techniques are then applied to solve the resulting CSP.
2. Generating a set of all possible CSPs from a *Conditional CSP*. Classical CSP techniques are then applied to solve each CSP.
3. Solving a *Conditional CSP* directly using constraint propagation.

To improve the performance of the third technique above, the constraint propagation methods *Forward Checking (FC)* and *Maintaining Arc Consistency (MAC)* are applied during the search [3]. Even though solving a *Conditional CSP* directly does not provide the best performance in time, this method is more flexible and time efficient when we are dealing with dynamic problems. Indeed, in the case of the second method, a small change of an activity constraint causes the regeneration of all possible sets of standard CSPs. Moreover, method 2 suffers from the waste of memory space needed to store many generated CSPs that will not be considered during search. In [2] *Composite CSPs* have extended the traditional CSP framework by including the combination of three new CSP paradigms: *Meta CSPs*, *Hierarchical Domain CSPs*, and *Dynamic CSPs*. In a composite CSP, the variable values can be entire sub CSPs. A domain can be a set of variables instead of atomic values (as it is the case in the traditional CSP). In configuration problems and planning, it is often obvious to organize the domains of variable values in a hierarchical manner. Jónsson and Frank [8] proposed a general framework using procedural constraints for solving dynamic CSPs. This framework has been extended to a new paradigm called Constraint-Based Attribute and Interval Planning (CAIP) for representing and reasoning about plans [9]. CAIP and its implementation, the EUROPA system, enable the description of planning domains with time, resources, concurrent activities, disjunctive preconditions and conditional constraints. The main difference, comparing to the formalisms we described earlier, is that in this latter framework [8] the set of constraints, variables and their possible values do not need to be enumerated beforehand which gives a more general definition of dynamic CSPs. Note that the definition of dynamic CSPs in [8] is also more general than the one in [7] since in this latter work variable domains are predetermined. Finally, in [10], Tsamardinos *et al* propose

the Conditional Temporal Problem (CTP) formalism for Conditional Planning under temporal constraints. This model extends the well known qualitative temporal network proposed in [11] by adding instantaneous events (called observation nodes) representing conditional constraints.

In this paper we combine the features of *Conditional CSPs* and *Composite CSPs* in a unique framework that we call *Conditional and Composite CSPs (CCCSPs)*. Our framework allows the representation of constraint problems where all the information corresponding to any possible change are known a priori. Indeed these latter information are added to the problem to solve in a dynamic manner, during the resolution process, via conditional (or activity) constraints and composite variables. A composite variable is a variable whose possible values are CSP variables. In other words this allows us to represent disjunctive variables where only one will be added to the problem to solve.

An activity constraint has the following form  $X_1 \wedge \dots \wedge X_p \xrightarrow{\text{condition}} Y$  where  $X_1, \dots, X_p$  and  $Y$  are variables (can be composite). This activity constraint will activate  $Y$  ( $Y$  will be added to the problem to solve) if  $X_1 \wedge \dots \wedge X_p$  are active (currently present in the problem to solve) and *condition* holds between these variables. In order to solve the CCCSP, we propose two methods that are respectively based on constraint propagation and Stochastic Local Search (SLS). The goal of the constraint propagation method is to overcome, in practice, the difficulty due to the exponential search space of the possible CSPs generated by the CCCSP to solve and also the search space we consider when solving each CSP. Indeed, a CCCSP represents  $D^M$  possible CSPs where  $D$  is the domain size of the composite variables and  $M$  the number of composite variables. In the same way as reported in [1,3], we use constraint propagation in order to detect earlier later failure. This will allow us to discard at the early stage any subset containing conflicting variables. The method based on constraint propagation is an exact technique that guarantees a complete solution. The method suffers however from its exponential time cost as we will notice by the experimental results we present in this paper. In many real-life applications where the execution time is an issue, an alternative will be to trade the execution time for the quality of the solution returned (number of solved constraints). This can be done by applying approximation methods such as local search and where the quality of the solution returned is proportional to the running time. Basically, the method we propose is based on Min-Conflict-Random-Walk (MCRW) [12] and consists of starting from a complete assignment of values to the different variables and iterates by improving at each step the quality of the assignment (number of solved constraints) until a complete solution is found or a maximum number of iterations is reached. The experimental study, we conducted on randomly generated CCCSPs demonstrates the efficiency of a variant of the MAC strategy (that we call MAC+) over the other constraint propagation techniques. We will also show that MAC+ outperform the SLS method MCRW for highly consistent CCCSPs. MCRW is however the procedure of choice for under constrained and middle constrained problems and also for highly constrained problems if we trade search time for the quality of the solution returned (number of solved constraints).

The rest of the paper is organized as follow. In the next section we will introduce our CCCSP model. Sections 3 and 4 are then respectively dedicated to the constraint propagation and SLS techniques we propose for solving CCCSPs. In Section 5 we

present the experimental study evaluating and comparing the methods we propose on randomly generated CCCSPs. Concluding remarks are finally listed in Section 6.

## 2 CCCSPs

A Conditional and Composite Constraint Satisfaction Problem (CCCSP) is a tuple  $\langle X, D_X, Y, D_Y, IV, C, A \rangle$ , where :

- $X = \{x_1, \dots, x_n\}$  is a finite set of variables.
- $D_X = \{D_{x_1}, \dots, D_{x_n}\}$  is the set of domains of the variables. Each domain  $D_{x_i}$  contains the possible values that  $x_i$  can take.
- $Y = \{y_1, \dots, y_m\}$  is the finite set of composite variables.
- $D_Y = \{D_{y_1}, \dots, D_{y_m}\}$  is the set of domains of the composite variables. Each domain  $D_{y_i}$  is the set of variables that the composite variable  $y_i$  can take.
- $IV$  is the set of initial variables (including composite variables):  $IV \subseteq X \cup Y$ .
- $C = \{C_1, \dots, C_p\}$  is the set of *compatibility constraints*. Each compatibility constraint is a binary relation between variables in case these latter variables are not composite, or a set of binary relations if at least one of the two variables involved is composite.
- $A$  is the set of *activity constraints*. Each activity constraint has the following form where  $Z_1, \dots, Z_p$  and  $T$  are variables (can be composite).  $Z_1 \wedge \dots \wedge Z_p \xrightarrow{\text{condition}} T$ . This activity constraint will activate  $T$  if  $Z_1, \dots, Z_p$  are active and *condition* holds on these variables. *condition* can be, for example, the assignment of particular values to the variables  $Z_1, \dots, Z_p$ .

We will use the following example to illustrate the CCCSP model.

### Example

*The goal of this configuration problem is to estimate the price of a new vehicle given a list of specifications. Figure 1 provides a complete list of features that the potential customer will choose from in order to build and price a new car. A solid line connecting a pair of nodes (circles) corresponds to a given constraint between the corresponding features. For instance, the constraint match color between Exterior and Windows contains all the pair of colors that go together. A solid arrow with a diamond corresponds to an activity constraint. For example, the choice of limited edition will activate the corresponding sunroof option. Dash lines connect the components to the class to which they belong. An OR connection corresponds to a list of features where only one will be selected by the customer. After the customer selects all the features, one or more consistent scenarios will be listed. The customer can also specify a maximum value of the price such that only scenarios with price less than this value will be displayed.*

Figure 2 illustrates the CCCSP model corresponding to the above example. There are 1 composite and 10 single variables (Model). The domain of the variables Exterior ,

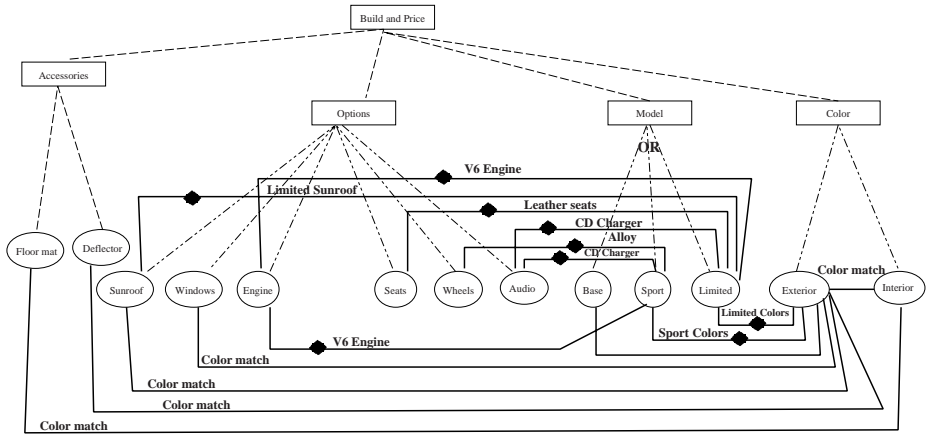


Fig. 1. The Build and Price Configuration Problem

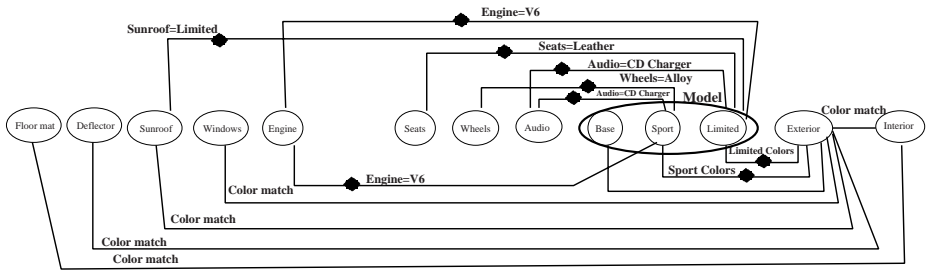


Fig. 2. CCCSP corresponding to the Build and Price Configuration Problem

Interior, Floor mat and Deflector contains a list of possible colors for each item. The domain of the other variables contains the different options corresponding to each feature. Each activity constraint is labelled by the activation condition and the constraint that will be added if the activation condition is true. For instance, if the composite variable Model is assigned the value Sport then the variable Engine will be activated and the constraint between Sport and Engine is that this latter should be assigned the value V6 from its domain.

### 3 Constraint Propagation for CCCSPs

Different methods for solving conditional CSPs have been reported in the literature [6,1,3,4]. In [6], all possible CSPs are first generated from the CCSP to solve. CSP techniques are then used on the generated CSPs in order to look for a possible solution. Dependencies between the activity constraints are considered in order to generate a directed a-cyclic graph (DAG), where the root node corresponds to the set of initially active variables. Activity constraints are applied during the derivation of one total order

from the partial order given by the resulting DAG. In [1,3] resolution methods have been proposed and are directly applied on CCSPs. Maintaining arc consistency (MAC) is used to prune inconsistent branches by removing inconsistent values during the search [3]. The solving method starts by instantiating the active variables. For each active variable instantiation, the algorithm first checks the compatibility constraints and then activates the activity constraints. The method will then enforce look-ahead consistency (through arc consistency) along the compatibility constraints and prunes inconsistent values from the domains of future variables. When activity constraints come into play, newly activated variables are added to the set of future variables. MAC is then applied to the set of all active variables. In [4,3], a CCSP is reformulated into an equivalent standard CSP. A special value “null” is added to the domains of all the variables which are not initially active. A variable instantiation with “null” indicates that the variable does not participate in the problem resolution. The CCSP is transformed into a CSP by including the “null” values. The disadvantage is that, in a large constraint problem, all variables and all constraints are taken into account simultaneously even if some are not relevant to the problem at hand.

In the above methods, backtrack search is used for both the generation of possible CSPs and the search for a solution in each of the generated CSPs. Thus, these methods require an exponential time for generating the different CSPs and an exponential time for searching a solution in each generated CSP. Moreover these methods are limited to handle only activity constraints. The other problem of the above methods is the redundant work done when checking at each time the consistency of the same set of variables (subset of a given generated CSP). The goal of the constraint propagation method we propose for solving CCCSPs is to overcome, in practice, the difficulty due to the exponential search space of the possible CSPs generated by the CCCSP to solve and also the search space we consider when solving each CSP. In the same way as reported in [1,3], we use constraint propagation in order to detect earlier later failure. This will allow us to discard at the early stage any subset containing conflicting variables. The description of the method we propose is as follows.

1. The method starts with an initial problem containing a list of initially activated variables (including composite variables). Arc consistency is applied on the initial variables in order to reduce some inconsistent values which will reduce the size of the search space. If the initial problem is not arc consistent (in the case of an empty domain) then the method will stop. The CCCSP is inconsistent in this case.
2. Following the forward check principle [13], pick an active variable  $v$ , assign a value to it and perform arc consistency between this variable and the non assigned active variables. If one domain of the non assigned variables becomes empty then assign another value to  $v$  or backtrack to the previously assigned variable if there are no more values to assign to  $v$ . Activate any variable  $v'$  resulting from this assignment and perform arc consistency between  $v'$  and all the active variables. If arc inconsistency is detected then deactivate  $v'$  and choose another value for  $v$  (since the current assignment of  $v$  leads to an inconsistent CCCSP). If  $v$  is a composite variable then assign a variable to it (from its domain). Basically, this consists of replacing the composite variable with one variable  $x$  of its domain. We then assign a value to  $x$  and proceed as shown before except that we do not backtrack in case all values of



$x$  are explored. Instead, we will choose another variable from the domain of the composite variable  $v$  or backtrack to the previously assigned variable if all values of  $v$  have been explored. This process will continue until all the variables are assigned in which case we obtain a solution to the CCCSP. The arc consistency in the above two steps is enforced as shown in the four cases below. We will assume in the following that  $x_1$  and  $x_2$  are non composite variables while  $y_1$  and  $y_2$  are composite.

- (a) **The constraint is  $(x_1, x_2)$ .** Arc consistency [14] is applied here i.e. each value  $a$  of  $x_1$  should have a support in the domain of  $x_2$ .
- (b) **The constraint is  $(y_1, x_1)$ .** Each value  $a$ , from the domain of a given variable  $x$  within  $y_1$ , should have a support in the domain of  $x_1$ .
- (c) **The constraint is  $(x_1, y_1)$ .** Each value  $a$ , from the domain of  $x_1$ , should have a support in at least one domain of the variables within  $y_1$ .
- (d) **The constraint is  $(y_1, y_2)$ .** Apply case 2 between  $y_1$  and each variable  $x$  within  $y_2$ .

Using the above rules, we have implemented a new arc consistency algorithm for CCCSPs as shown in Figure 3. This algorithm is an extension of the well known AC-3 procedure [14,15,16].

```

REWISE( $D_i, D_j$ )
  REWISE  $\leftarrow$  false
  For each value  $a \in D_i$  do
    if not compatible( $a, b$ ) for any value  $b \in D_j$  then
      remove  $a$  from  $D_i$ 
      REWISE  $\leftarrow$  true
    end if
  end for

REWISE_COMP( $D_i, D_j$ )
  REWISE_COMP  $\leftarrow$  false
  if  $i$  is a single variable and  $j$  is a composite variable
     $D_{tmp} \leftarrow \emptyset$ 
    For each event  $k \in D_j$  do
       $D \leftarrow D_i - D_{tmp}$ 
      REWISE_COMP  $\leftarrow$  REWISE_COMP OR REWISE( $D, D_k$ )
       $D_{tmp} \leftarrow D_{tmp} \cup D$ 
    end for
     $D_i \leftarrow D_{tmp}$ 
  end if
  if  $i$  is a composite variable and  $j$  is a single variable
    For each event  $k \in D_i$  do
      REWISE_COMP  $\leftarrow$  REWISE_COMP OR REWISE( $D_k, D_j$ )
    end for
  end if
  if  $i$  and  $j$  are composite variables
    For each event  $k \in D_i$  do
      REWISE_COMP( $D_k, D_j$ )
    end for
  end if

AC-3-CCCSP
  Given a graph  $G = (X, U)$ 
   $Q \leftarrow \{(i, j) | i, j \in U\}$ 
  while  $Q \neq Nil$  do
     $Q \leftarrow Q - \{(i, j)\}$ 
    if  $i$  or  $j$  is composite variable
      if REWISE_COMP( $D_i, D_j$ ) then
         $Q \leftarrow Q \cup \{(k, i) | k \in U \text{ and } k \neq j\}$ 
      end if
    else if REWISE( $D_i, D_j$ ) then
       $Q \leftarrow Q \cup \{(k, i) | k \in U \text{ and } k \neq j\}$ 
    end if
  end if
end while

```

**Fig. 3.** AC-3 for CCCSPs

Like for general CSPs, variable and value ordering, during search, has a significant impact on the size of the explored space in the case of CCCSPs. For variable selection, we will follow the idea of choosing the most constrained variable first in the hope of triggering early failure. In the case of value selection, we start with the value that leads to an easiest to solve CCCSP first since our goal here is to find the first solution and that there is no preference on the solution obtained. More precisely, our variable and value selection policy works as follows.

1. The variables (simple and composite variables) are selected by decreasing order of the number of constraints they share with other variables. For a given variable  $x$ , this number (that we call degree of a variable) corresponds to the node degree (number of edges connected to the node) of the node corresponding to  $x$  in the constraint graph.
2. The degree of a composite variable  $x$  is equal to the minimum variable degree of all the variables within  $X$  domain.
3. If a variable  $x$  activates other variables then we add to its variable degree, the minimum number of constraints it can generate (activate) through the activity constraints.
4. For value selection, in the case of a composite variable  $x$ , select the simple variables, within the domain of  $x$ , by decreasing number of their degrees.
5. For a simple variable select the least constrained value first (the value that causes the activation of the minimum number of constraints).

## 4 MCRW for CCCSPs

The method we presented in the previous Section is an exact technique that guarantees a complete solution. The method suffers however from its exponential time cost as we will see in the next Section. In many real-life applications where the execution time is an issue, an alternative will be to trade the execution time for the quality of the solution returned (number of satisfied constraints). This can be done by applying approximation methods such as local search and where the quality of the solution returned is proportional to the running time. In this Section we will study the applicability of a local search technique based on the Min-Conflict-Random-Walk (MCRW) [12] algorithm for solving CCCSPs. Basically, the method consists of starting from a complete assignment of values to variables and iterates by improving at each step the quality of the assignment (number of satisfied constraints) until a complete solution is found or a maximum number of iterations is reached. Given the dynamic aspect of CCCSPs (some variables are added/removed dynamically during the resolution process) we propose the following algorithm based on MCRW for solving CCCSPs.

1. The algorithm starts with a random assignment of values to the initial variables. If the initial variable is composite then it will be replaced by one variable selected randomly from its domain. This latter variable will then be randomly assigned a value from its domain.
2. Activate any variable where the activating condition is true and randomly assign to it a value from its domain as shown in the previous step.

3. If a complete solution is not found and the maximum number of iterations is not reached, randomly select an active variable  $v$  and proceed with one of the following two cases.
  - If  $v$  belongs to the domain of a given composite variable  $y$  then select the pair  $\langle v, val \rangle$  that increases the quality of the current solution (number of solved constraints).  $v$  belongs here to the domain of  $y$  and  $val$  is a value of  $v$ 's domain,
  - otherwise, assign to  $v$  a value that increases the quality of the solution.
4. Deactivate any variable activated by the old assignment of  $v$  and goto 2.

## 5 Experimentation

In order to evaluate the methods we propose, we have performed experimental tests on randomly generated CCCSPs. The experiments are performed on a PC Pentium 4 computer running Linux. All the procedures are coded in C/C++. CCCSPs are build from CSPs randomly generated by the model RB proposed in [17]. This model has exact phase transition and the ability to generate asymptotically hard instances. Following the model RB, we generate each CSP instance as follows using the parameters  $n$ ,  $p$ ,  $\alpha$  and  $r$  where  $n$  is the number of variables,  $p$  ( $0 < p < 1$ ) is the constraint tightness, and  $r$  and  $\alpha$  ( $0 < \alpha < 1$ ) are two positive constants. 1) Select with repetition  $rn \ln n$  random constraints. Each random constraint is formed by selecting without repetition 2 of  $n$  variables. 2) For each constraint we uniformly select without repetition  $pd^k$  incompatible pairs of values, where  $d = n^\alpha$  is the domain size of each variable. Each CCCSP instance is then generated as follows.

1. Randomly generate a CSP with the parameters  $n$ ,  $p$ ,  $\alpha$  and  $r$  as shown above.
2. Generate  $N$  composite variables each containing  $D$  simple variables.
3. Select with repetition  $r[(n+N) \ln(n+N) - n \ln n]$  new random constraints (between the  $n+N$  variables), each formed by selecting without repetition 2 of the  $n+N$  variables. This will guarantee that the total number of constraints is  $r(n+N) \ln(n+N)$ . For each constraint we uniformly select without repetition  $pd^k$  incompatible pairs of values.
4. Select  $I(n+N)$  initial variables from  $n+N$  ( $0 < I < 1$ ).
5. Select  $a(nd+ND)$  activity constraints for each of the  $n+N-I(n+N)$  non initial variables ( $0 < a < 1$ ).

As demonstrated in [17], when the number of variables approaches infinity the phase transition occurs when the constraint tightness  $p = 1 - e^{-\frac{d}{r}}$ . Thus the phase transition is an asymptotic phenomenon since, only for infinite number of variables, we can have sharp phase transitions. In addition, the number of variables and constraints of the possible CSPs, each CCCSP contains, is slightly different from the one of the CCCSP they are generated from. The tests we have performed compare the following four propagation strategies.

- 1) Forward Check (FC).** This is the strategy we have described in Section 3 which consists of maintaining arc consistency, during the search, between the current variable (the variable that we are assigning a value) and the future active variables (variables not yet assigned) sharing a constraint with the current variable.

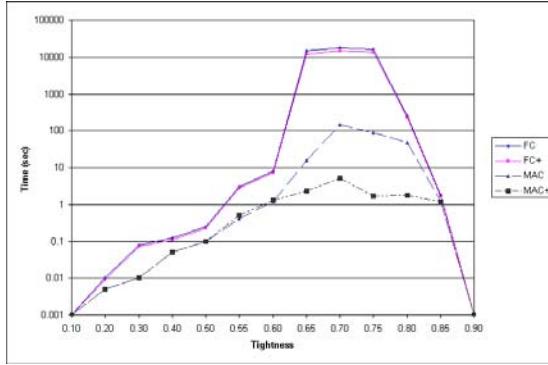


Fig. 4. Comparative tests on random CCCSPs

- 2) **Maintaining Arc Consistency (MAC).** This strategy maintains a full arc consistency on the current and future active variables.
- 3) **FC+.** Same as FC except that the applicability of the arc consistency is extended to non active variables as well.
- 4) **MAC+.** Same as MAC except that the applicability of the arc consistency is extended to non active variables as well.

Figure 4 presents the results of comparative tests performed on random consistent CCCSPs generated with the following parameters:  $n = 140$ ,  $N = 10$ ,  $D = 5$ ,  $\alpha = 0.8$ ,  $I = 0.8$ ,  $a = 0.2$  and  $r = 0.6$ . As mentioned earlier, the phase transition can be computed as follows:  $p = 1 - e^{-\frac{\alpha}{r}} = 1 - e^{-\frac{0.8}{0.6}} = 0.73$ . Thus, consistent instances are those with the tightness less than 0.73. For each test (corresponding to a particular tightness value  $p$ ), each of the four methods is executed on 100 instances and the average running time in seconds is taken. All the methods have similar running times in the case of under constrained problems. Indeed, in this particular case the extra effort done by MAC and MAC+ does not remove much of the inconsistent values and thus does not improve the overall running time to find a solution. However when we move toward the phase transition the extra work performed by MAC and especially MAC+ starts to pay off. At the phase transition MAC+ is almost 10,000 times faster than FC and FC+; and 100 times faster than MAC. In general MAC+ is the best method for solving random CCCSPs.

In order to compare MAC+ to the MCRW method, we run this latter technique to the same problem instances described above. Since MCRW is an approximation method, in case it does not find a complete solution for a given instance we report the quality (percentage of solved constraints) of the best solution obtained and the time it took to get this quality. Note also that we only consider consistent instances (more specifically those with tightness less than or equal to 0.7). The test results are reported in table 1. For each test, each method is executed on 100 instances and the average running time in seconds is taken. As we can see in the table, MCRW outperforms MAC+ for under and middle constrained problems. Indeed, in the case of under constrained problems, for example, the solution is obtained in the case of MCRW after a couple of random

**Table 1.** Comparative tests on random CCTCSPs

Tightness	MCRW		MAC+
	Time	success(%)	
0.1	0	100	0
0.2	0	100	0.01
0.3	0	100	0.01
0.4	0.01	100	0.05
0.5	0.08	100	0.1
0.6	0.3	80	1.3
0.7	0.4	70	5.2

assignments. However when we approach the phase transition, the random search is affected by the change of the constraint network at each iteration. Indeed, each time an assignment is reconsidered it usually results in deactivating several variables and activating others. MCRW has then to restart with this new configuration. Note that while MCRW does not solve completely highly constrained problems, it is still a method of choice in case we want to trade search time for the quality of the solution returned. As we can see in the table, in the case where the tightness is equal to 0.7 for example, we can decide to get the incomplete solution (solving 70% of the constraints) within 0.4 seconds with MCRW instead of waiting 5.2 seconds to get a complete one. Trading search time for the quality of the solution can be very relevant for reactive and real time applications where an answer is needed within a given deadline.

## 6 Conclusion

We have presented in this paper a CSP based framework for representing and managing compatibility constraints, activity constraints and composite variables with a unique constraint network that we call Conditional Composite Constraint Satisfaction Problem (CCCSP). Solving a CCCSP consists of finding a solution for one of its possible CSPs. This requires an algorithm with  $O(D^N d^M)$  time cost where  $N, D, M$  and  $d$  are respectively the number of non composite variables and their domain size, the number of composite variables and their domain size. In order to overcome this difficulty in practice, we have proposed two methods respectively based on constraint propagation and Stochastic Local Search (SLS). Constraint propagation prevents earlier later failure which improves, in practice, the performance in time of the backtrack search. On the other hand, because of its polynomial time cost, the SLS method has better time performance than constraint propagation but does not always guarantee a complete solution. The experimental study, we conducted on randomly generated CCCSPs demonstrates the efficiency of a variant of the MAC strategy (that we call MAC+) over the other constraint propagation techniques. We will also show that MAC+ outperforms the SLS method MCRW for highly consistent CCCSPs. MCRW is however the procedure of choice for under constrained and middle constrained problems and also for highly constrained problems if we trade search time for the quality of the solution returned (number of solved constraints).

## References

1. Mittal, S., Falkenhainer, B.: Dynamic constraint satisfaction problems. In: Proceedings of the 8th National Conference on Artificial Intelligence, Boston, MA, AAAI Press (August 1990) 25–32
2. Sabin, D., Freuder, E.C.: Configuration as composite constraint satisfaction. In Luger, G.F., ed.: Proceedings of the (1st) Artificial Intelligence and Manufacturing Research Planning Workshop, AAAI Press (1996) 153–161
3. D. Sabin, E.C.F., Wallace, R.J.: Greater efficiency for conditional constraint satisfaction. Proc., Ninth International Conference on Principles and Practice of, Constraint Programming - CP 2003 **2833** (2003) 649–663
4. Gelle, E.: On the generation of locally consistent solution spaces in mixed dynamic constraint problems. Ph.D.thesis **1826** (1998) 101–140
5. Gelle, E., Faltings, B.: Solving mixed and conditional constraint satisfaction problems. Constraints **8** (2003) 107–141
6. Gelle, E., Sabin, M.: Solving methods for conditional constraint satisfaction. In: IJCAI-03, Workshop on Configuration, Acapulco, Mexico (2003)
7. Dechter, R., Dechter, A.: Belief maintenance in dynamic constraint networks. In: 7th National Conference on Artificial Intelligence, St Paul (1988) 37–42
8. Jónsson, A.K., Frank, J.: A framework for dynamic constraint reasoning using procedural constraints. In: ECAI 2000. (2000) 93–97
9. J. Frank, A.K.J.: Constraint-based attribute and interval planning. Constraints **8**(4) (2003) 339–364
10. Tsamardinos, I., Vidal, T., Pollack, M.E.: CTP: A New Constraint-Based Formalism for Conditional Temporal Planning. Constraints **8**(4) (2003) 365–388
11. Dechter, R., Meiri, I., Pearl, J.: Temporal Constraint Networks. Artificial Intelligence **49** (1991) 61–95
12. Selman, B., Kautz, H.: Domain-independent extensions to gsat: Solving large structured satisfiability problems. In: IJCAI-93. (1993) 290–295
13. Haralick, R., Elliott, G.: Increasing tree search efficiency for Constraint Satisfaction Problems. Artificial Intelligence **14** (1980) 263–313
14. Mackworth, A.K.: Consistency in networks of relations. Artificial Intelligence **8** (1977) 99–118
15. Bessière, C., Régin, J.C.: Refining the basic constraint propagation algorithm. In: Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01), Seattle, WA (2001) 309–315
16. Zhang, Y., Yap, R.H.C.: Making ac-3 an optimal algorithm. In: Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01), Seattle, WA (2001) 316–321
17. Xu, K., Li, W.: Exact Phase Transitions in Random Constraint Satisfaction Problems. Journal of Artificial Intelligence Research **12** (2000) 93–103

# Multiagent Constraint Satisfaction with Multiply Sectioned Constraint Networks

Y. Xiang and W. Zhang

University of Guelph, Canada

**Abstract.** Variables and constraints in problem domains are often distributed. These distributed constraint satisfaction problems (DCSPs) lend themselves to multiagent solutions. Most existing algorithms for DCSPs are extensions of centralized backtracking or iterative improvement with breakout. Their worst case complexity is exponential. On the other hand, directional consistency based algorithms solve centralized CSPs efficiently if primal graph density is bounded. No known multiagent algorithms solve DCSPs with the same efficiency. We propose the first such algorithm and show that it is sound and complete.

## 1 Introduction

Many practical problems can be solved as constraint satisfaction problems (CSPs). Often, the variables and constraints in the problem domain are naturally distributed, spatially, cognitively, or otherwise. These distributed CSPs (DCSPs) [13] lend themselves naturally to solutions using multiagent systems.

Most existing algorithms for solving DCSPs are extensions of centralized algorithms based on backtracking or iterative improvement with breakout [13,11,14,5,7,9,8]. Their worst case complexity is exponential. Another class of algorithms [12] is based on truth maintenance, e.g., DATMS [4]. The complexity of truth maintenance problem is at least NP-hard [6].

On the other hand, directional consistency based algorithms [2,3] solve centralized CSPs efficiently if the density of the primal graph (measured by *tree width*) is upper-bounded. To the best of our knowledge, no existing multiagent algorithms solve DCSPs with the same efficiency. In this work, we propose the first such algorithm. We present formally an multiagent representation of DCSPs. We prove soundness and completeness of the algorithm and illustrate with a detailed example. Due to space limitations, however, we omit proofs.

## 2 Background

CSPs are formally modeled as constraint networks. A *constraint network* (CN)  $\mathcal{R}$  is a pair  $\mathcal{R} = (V, A)$ .  $V$  is an non-empty set of discrete variables, called *domain*. Each variable  $v \in V$  has a finite space  $D_v$ . The space of a subset  $X \subset V$  is the Cartesian product of spaces of variables in  $X$  and is denoted by  $D_X$ . Each  $\underline{x} \in D_X$  is a configuration of  $X$ .  $A$  is an non-empty set of constraints. Each

constraint is a relation  $R_X \subseteq D_X$ , where  $X \subset V$  is the *scope* of the constraint. The union of scopes of all constraints covers the domain, i.e.,  $\cup_{R_X \in \Lambda} X = V$ .

A configuration  $\underline{x} \in D_X$  satisfies a constraint  $R_X$  if  $\underline{x} \in R_X$ . Otherwise, it *violates* the constraint. The *projection* of configuration  $\underline{x}$  to  $Y \subset X$  is denoted by  $\pi_Y(\underline{x})$  and the projection of relation  $R_X$  to  $Y \subset X$  is denoted by  $\pi_Y(R_X)$ . Configuration  $\underline{x}$  is *consistent* or *legal* if it satisfies every constraint  $R_Y$  such that  $Y \subseteq X$ . A *solution* to CN  $\mathcal{R}$  is a consistent configuration over  $V$ . Formally, the set of all solutions, called the *solution set*, of  $\mathcal{R}$  is the relation  $\bowtie_{R \in \Lambda} R$ , where  $\bowtie$  refers to relational operator *natural join*.  $\mathcal{R}$  is *consistent* iff  $\bowtie_{R \in \Lambda} R \neq \emptyset$ .

The dependence structure of  $\mathcal{R}$  can be depicted by a *primal graph*  $G = (V, E)$ , where each node is labeled by a variable  $v \in V$  and each link  $\langle u, v \rangle \in E$  connects nodes  $u, v$  if there exists a constraint  $R_X \in \Lambda$  such that  $u, v \in X$ .  $\mathcal{R}$  can be solved through an alternative dependence structure compiled from its primal graph. A *cluster* is a subset of  $V$ . A *cluster tree* connects a set of clusters into a tree. Each link, called a *separator*, connects two clusters whose intersection  $S \neq \emptyset$ , and is labeled by  $S$ . A cluster tree is a *junction tree* (JT) if the intersection of each pair of clusters is a subset of every separator on the path between them. Details on how to compile a graph into a JT can be found in [10].

For DCSP, we assume that variables and constraints are distributed among multiple agents such that each agent is in charge of a CN. We introduce concepts for description of primal graphs from multiple CNs to be used later. Let  $G_i = (V_i, E_i)$  ( $i = 0, 1$ ) be two graphs.  $G_0$  and  $G_1$  are *graph-consistent* if subgraphs of  $G_0$  and  $G_1$  spanned by  $V_0 \cap V_1$  are identical. Given two graph-consistent graphs  $G_i = (V_i, E_i)$  ( $i = 0, 1$ ), the graph  $G = (V_0 \cup V_1, E_0 \cup E_1)$  is the *union* of  $G_0$  and  $G_1$ , denoted by  $G = G_0 \cup G_1$ . Given a graph  $G = (V, E)$ , a partition of  $V$  into  $V_0$  and  $V_1$  such that  $V_0 \cup V_1 = V$  and  $V_0 \cap V_1 \neq \emptyset$ , and subgraphs  $G_i$  ( $i = 0, 1$ ) of  $G$  spanned by  $V_i$ ,  $G$  is said to be *sectioned* into  $G_0$  and  $G_1$ .

### 3 Solving CSP with Junction Tree Representation

The method for solving centralized CSPs is attributed to Dechter and Pearl [2,3,1]. Our work extends theirs to multiagent systems. We review the method so that its components can be directly referenced later in presenting our extension. Our formulation, however, is not necessarily identical to that in the references.

Given CN  $\mathcal{R} = (V, \Lambda)$  and its primal graph  $G$ , first, compile  $G$  into a JT  $T$ . Second, for each constraint  $R_X$  in  $\Lambda$ , assign  $R_X$  to a cluster  $Q$  in  $T$  such that  $X \subseteq Q$ . Third, for each cluster  $Q$  in  $T$ , replace the set  $A_Q$  of constraints assigned to it by a single constraint  $R_Q = U_Q \bowtie_{R \in A_Q} R$ , where  $U_Q$  is a universal relation over  $Q$  (containing every configuration of  $Q$ ). Let  $\Lambda'$  denotes the set of new constraints one per cluster of  $T$ . Note that  $\Lambda'$  is simply a grouping of  $\Lambda$ . Finally, let each cluster in  $T$  be a variable and its space be configurations in the relation associated with the cluster. For each pair of adjacent clusters  $Q$  and  $C$  with separator  $S$ , impose the *implicit constraint* between  $Q$  and  $C$ :  $\pi_S(\underline{q}) = \pi_S(\underline{c})$ , where  $\underline{q}$  is a configuration of  $Q$  and  $\underline{c}$  is a configuration of  $C$ . The triple  $(V, T, \Lambda')$  is the *JT representation* of  $\mathcal{R}$  and its solution set is  $\bowtie_{R \in \Lambda'} R$ .



Note that  $\Lambda'$  does not include implicit constraints since they simply allows  $\bowtie$  operation to be well defined. Note also that  $(V, T, \Lambda')$  is equivalent to a binary CN. Proposition 1 below establishes the key property of the JT representation and plays an important role in analysis of our method.

**Proposition 1.** *Let  $(V, \Lambda)$  be a CN and  $(V, T, \Lambda')$  be its JT representation. The solution set of  $(V, \Lambda)$  and that of  $(V, T, \Lambda')$  are identical.*

The complexity of the compilation is  $O(|\Lambda| k^q)$ , where  $k$  binds the space for variables in  $V$  and  $q$  binds the size for clusters in  $T$ .  $(V, T, \Lambda')$  can be solved based on directional arc-consistency. Given two clusters  $Q$  and  $C$  with  $S = Q \cap C$ , configurations  $\underline{q}$  and  $\underline{c}$  are *consistent* if  $\pi_S(\underline{q}) = \pi_S(\underline{c})$ . A cluster  $Q$  in  $T$  is *consistent relative to* an adjacent cluster  $C$  if, for each configuration in  $R_Q$ , there exists a consistent configuration in  $R_C$ . Let  $Q^*$  be any cluster in  $T$ . Given  $Q^*$ ,  $T$  can be viewed as a tree rooted at  $Q^*$  and each two adjacent clusters form a parent-child pair.  $(V, T, \Lambda')$  is *directional arc-consistent* relative to a root cluster  $Q^*$  if for every pair of clusters  $Q$  and  $C$ , where  $Q$  is the parent of  $C$ ,  $Q$  is consistent relative to  $C$ .

The following object oriented algorithm is activated at each cluster in  $T$  by a *caller*, which is either an adjacent cluster or the object  $T$ . After it is called in  $Q^*$  by  $T$ ,  $(V, T, \Lambda')$  is directional arc-consistent relative to  $Q^*$ .

**Algorithm 1 (CollectSeparatorConstraint).** *When caller calls in cluster  $Q$ , it does the following:*

*$Q$  calls CollectSeparatorConstraint in each adjacent cluster  $C$  except caller;  
for each cluster  $C$  (whose separator with  $Q$  is  $S$ ),*

*$Q$  receives from  $C$  a constraint  $R_S$ ;*

*if  $R_S = \emptyset$ ,  $Q$  sends  $\emptyset$  to caller and halts;*

*$Q$  assigns  $R_Q = R_Q \bowtie R_S$ ;*

*if caller is a cluster (whose separator with  $Q$  is  $S'$ ),  $Q$  sends  $\pi_{S'}(R_Q)$  to caller;*

The complexity of CollectSeparatorConstraint is  $O(t l^2)$ , where  $t$  is the number of clusters in  $T$  and  $l$  binds the size of relation in each cluster. After CollectSeparatorConstraint is called in  $Q^*$ , if  $\emptyset$  is returned, the CN is inconsistent. Otherwise,  $(V, T, \Lambda')$  can be solved by  $T$  calling the following algorithm in  $Q^*$ . It will then be called recursively at each cluster.

**Algorithm 2 (DistributeSeparatorSolution).** *When caller calls in cluster  $Q$ , it does the following:*

*if caller is a cluster (whose separator with  $Q$  is  $S$ ),*

*$Q$  receives from caller a constraint  $R_S$  of a single configuration;*

*$Q$  assigns  $R_Q = \{\underline{q}\}$ , where  $\underline{q} \in R_Q \bowtie R_S$ ;*

*else  $Q$  removes all configurations in  $R_Q$  except one;*

*for each adjacent cluster  $C$  (whose separator with  $Q$  is  $S'$ ) except caller;*

*$Q$  calls DistributeSeparatorSolution in  $C$  with  $\pi_{S'}(R_Q)$ ;*

After `DistributeSeparatorSolution` is called in  $Q^*$ , the solution to  $(V, T, A')$  can be obtained by retrieving  $R_Q$  from each cluster  $Q$  and joining them together.

`CollectSeparatorConstraint` above only achieves directional arc-consistency. A parent cluster  $Q$  (relative to a root) is consistent relative to a child cluster  $C$ , but  $C$  may not be consistent relative to  $Q$ . This is possible because the constraint  $R_S$  sent from  $C$  to  $Q$  during `CollectSeparatorConstraint` may contain a configuration  $\underline{s}$  such that no configuration  $\underline{q}$  in  $R_Q$  satisfies  $\pi_S(\underline{q}) = \underline{s}$ . Adjacent clusters  $Q$  and  $C$  are *consistent* if  $Q$  is consistent relative to  $C$  and vice versa.  $(V, T, A')$  is *full arc-consistent* if every pair of adjacent clusters is consistent. Full full arc-consistency is not needed to solve  $(V, T, A')$  as shown above. However, it is necessary for solving DCSPs as will be seen.

The following object oriented algorithm can be performed after `CollectSeparatorConstraint` to make a JT full arc-consistent.

**Algorithm 3 (`DistributeSeparatorConstraint`).** *When caller calls in cluster  $C$ , it does the following:*

*if caller is a cluster (whose separator with  $C$  is  $S$ ),*  
*$C$  receives from caller a constraint  $R_S$ ;*  
*$C$  assigns  $R_C = R_C \bowtie R_S$ ;*  
*for each adjacent cluster  $Q$  (whose separator with  $C$  is  $S'$ ) except caller,*  
*$C$  calls `DistributeSeparatorConstraint` in  $Q$  with  $\pi_{S'}(R_C)$ ;*

The following algorithm combines `CollectSeparatorConstraint` and `DistributeSeparatorConstraint`.

**Algorithm 4 (`UnifyConstraint`).** *Choose a cluster  $Q^*$  arbitrarily and call `CollectSeparatorConstraint` in  $Q^*$ . If  $Q^*$  returns  $\emptyset$ , return false. Otherwise, call `DistributeSeparatorConstraint` in  $Q^*$  and return true upon completion.*

After `UnifyConstraint`, a JT is full arc-consistent as summarized below.

**Proposition 2.** *Let  $(V, T, A')$  be the JT representation of a CN. The CN is inconsistent iff `UnifyConstraint` returns false. Otherwise, `UnifyConstraint` returns true and the JT is full arc-consistent.*

## 4 Multiply Sectioned Constraint Network

A DCSP involves a large problem domain where variables and constraints are distributed. We solve a DCSP with a multiagent system, where each agent is in charge of a subset of variables and constraints. To ensure that computation is sound and complete as well as efficient, partition of variables and constraints among agents needs to satisfy certain conditions. We model a DCSP as an multiply sectioned constraint network (MSCN) which specifies these conditions formally.

**Definition 1.** *From a set of CNs  $\{\mathcal{R}_i = (V_i, A_i)\}$  (each called a **subnet**), an MSCN  $\mathcal{R}$  is defined as a pair  $\mathcal{R} = (V, A)$ , where  $V = \bigcup_i V_i$  is the domain (with*

each  $V_i$  called a **subdomain**) and  $\Lambda = \bigcup_i \Lambda_i$  is the set of constraints, such that the following holds: (1) A JT exists with  $\{V_i\}$  as the set of clusters. (2) For any two subnets  $\mathcal{R}_i$  and  $\mathcal{R}_j$  such that  $V_i \cap V_j \neq \emptyset$ , their primal graphs are graph-consistent. The solution set of  $\mathcal{R}$  is  $\bowtie_i (\bowtie_{R \in \Lambda_i} R)$ .

This concise definition has a number of implications: First of all, although there is no mention of agents in the definition, we assume that each subnet  $\mathcal{R}_i$  is embodied by a unique agent  $A_i$  who is in charge of subdomain  $V_i$ . Hence, a variable shared by two subnets are *public* to the corresponding agents and a variable unique in a subnet is *private*.

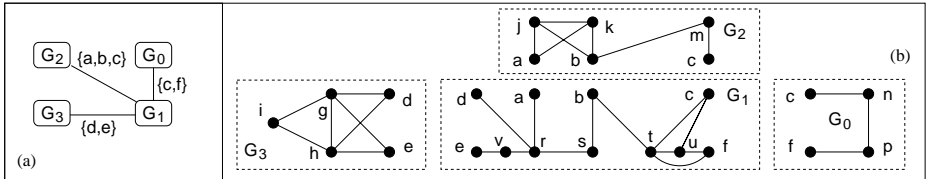
Second, domain partition is required to satisfy the connectivity condition (a JT is connected). That is, for any two subdomains  $V_i$  and  $V_j$ , there exists a sequence of subdomains such that every two adjacent in the sequence share some variables. This restriction implies that each subnet is relevant to the partial solution in each other subnet.

Third, domain partition is required to satisfy the JT condition. Although a natural domain partition may not satisfy this condition, it can be enforced by making limited private variables public. Agents  $A_i$  and  $A_j$  are said to be *adjacent* if  $V_i$  and  $V_j$  are adjacent in the JT.

Fourth, primal graphs are required to be graph-consistent. This means that every constraint over public variables in one subnet must be contained in every other subnet that share these variables. We assume that this condition is enforced by communicating any constraint over public variables to other agents in a pre-processing. Similarly, if a constraint  $R_Z$  has a scope  $Z = X \cup Y$ , where  $X \cap Y = \emptyset$ ,  $X$  is public, and  $Y$  is private, we assume that the constraint  $\pi_X(R_Z)$  has been communicated to the other agent. The condition essentially ties variable sharing between subnets with constraint sharing.

Fifth, as each subnet uniquely defines its primal graph and these primal graphs are graph-consistent, the collection of primal graphs from all subnets defines a multiply sectioned primal graph over the domain, and hence the name MSCN.

Sixth, although an MSCN may admit multiple JTs (condition (1)), one of them, referred to as the *hypertree*, is agreed upon by all agents and governs agent communication. That is, if  $A_i$  and  $A_j$  are adjacent in the hypertree, then they can communicate directly. We refer to each cluster  $V_i$  in the hypertree as a *hypernode*, and associate the hypernode with subnet  $\mathcal{R}_i$  and agent  $A_i$ . Hence, the



**Fig. 1.** The hypertree (a) and primal graphs (b) of an MSCN. Each link in (b) represents a  $\neq$  constraint.

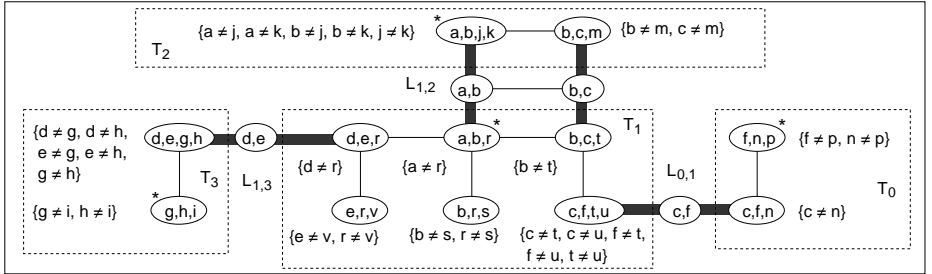
hypertree is the agent organization. If  $A_i$  and  $A_j$  are adjacent in the organization, we refer to  $V_i \cap V_j$  as their *agent interface*.

Finally, joining a relation multiple times to another relation has the effect of exactly once. Hence, communicating constraints over public variables, as mentioned above, has no impact on the solution set.

Fig. 1 shows a distributed map coloring problem as an example MSCN. The primal graphs of subnets are shown in (b) and the hypertree is shown in (a). The space of each variable contains three colors which we denote simply by  $\{0, 1, 2\}$ .

## 5 Linked Junction Forest Representation of MSCN

To extend JT based solution of CNs to MSCNs, we compile MSCNs to a runtime representation. Exploring structural similarity between constraint reasoning and probabilistic reasoning, we adopt key steps in structure compilation of multiply sectioned Bayesian networks (MSBNs)[10]: cooperative triangulation, local JT construction, and linkage tree (LT) construction. Formal specification in the context of MSBNs can be found in the reference. Outcome of structure compilation for the MSCN in Fig. 1 is shown in Fig. 2.



**Fig. 2.** The linked junction forest compiled from MSCN in Fig. 1. The constraint assigned to each cluster is shown in  $\{\}$ .

Each subnet is compiled into a JT, e.g., subnet  $G_1$  is compiled into  $T_1$ . Each agent interface is compiled into a LT, e.g., the agent interface between  $A_1$  and  $A_2$  is compiled into LT  $L_{1,2}$  which consists of two clusters. Each cluster in  $L_{1,2}$  is referred to as a *linkage*, e.g.,  $\{b, c\}$ . Each linkage has two *host clusters* one in each JT it links. For instance, linkage  $\{b, c\}$  has host cluster  $\{b, c, t\}$  in  $T_1$  and host cluster  $\{b, c, m\}$  in  $T_2$ .

After the structure compilation, each agent  $A_i$  assigns constraints in  $A_i$  to clusters in  $T_i$  as follows: For each constraint  $R_X$  in  $A_i$ , assign  $R_X$  to a cluster  $Q$  in  $T_i$  such that  $X \subseteq Q$ . After assignment, for each cluster  $Q$  in  $T_i$ ,  $A_i$  replaces the set  $A_Q$  of constraints assigned to it by a single constraint  $R_Q = U_Q \bowtie_{R \in A_Q} R$ , where  $U_Q$  is the universal relation over  $Q$ .

Let each cluster in  $T_i$  be a variable and its space be configurations in the relation associated with the cluster. For each pair of adjacent clusters  $Q$  and  $C$

with separator  $S$ , let the *implicit constraint* between  $Q$  and  $C$  be  $\pi_S(\underline{q}) = \pi_S(\underline{c})$ , where  $\underline{q}$  is a configuration of  $Q$  and  $\underline{c}$  is a configuration of  $C$ . For instance, constraint between clusters  $\{c, f, t, u\}$  and  $\{b, c, t\}$  in  $T_1$  requires their configurations to have the same value over  $c$  and  $t$ . The similar implicit constraint is imposed relative to each linkage  $S$  and its two linkage hosts  $Q$  and  $C$ . For instance, constraint between linkage hosts  $\{b, c, t\}$  in  $T_1$  and  $\{b, c, m\}$  in  $T_2$  requires their configurations to have the same value over  $b$  and  $c$ .

Given an MSCN  $\mathcal{R} = (V = \bigcup_i V_i, \Lambda = \bigcup_i \Lambda_i)$ , the outcome of compilation is a tuple  $(V, T, L, \Lambda')$ , where  $T = \{T_i\}$  is a set of JTs each compiled from a subnet of  $\mathcal{R}$ , and  $L = \{L_{i,j}\}$  is a set of LTs one compiled from each pair of adjacent subnets on hypertree.  $\Lambda' = \{\Lambda'_i\}$  is a collection of sets. Each  $\Lambda'_i$  is a set of constraints one per cluster of  $T_i$ . We refer to  $(V, T, L, \Lambda')$  as the *linked junction forest representation* (LJF) of the MSCN. Again, we assume that agents are attached to LJF such that each  $T_i$  is embodied by  $A_i$ . The solution set of  $(V, T, L, \Lambda')$  is  $\bowtie_i (\bowtie_{R \in \Lambda'_i} R)$ . Note that  $\Lambda'$  does not include implicit constraints since they simply allow  $\bowtie$  operation to be well defined. The following theorem establishes an important property of the LJF. It follows from definitions of solution sets for MSCN and its LJF, as well as the composition of  $\Lambda'$ .

**Theorem 1.** *Let  $\mathcal{R} = (\bigcup_i V_i, \bigcup_i \Lambda_i)$  be an MSCN and  $\mathcal{F} = (V, T, L, \Lambda')$  be its LJF representation. Then,  $\mathcal{R}$  and  $\mathcal{F}$  have the same solution set.*

The compilation computation is dominated by the cooperative triangulation and local JT construction. The complexity is  $O(n \lambda k^q)$ , where  $n$  is the number of agents,  $\lambda$  bounds  $|A_i|$ ,  $k$  binds the space for variables in  $V$  and  $q$  binds the size for clusters in JTs in  $T$ .

## 6 Solving MSCN with LJF

To solve MSCN using its LJF, we extend directional arc-consistency to LJF. An agent  $A_i$  is *interface-consistent* relative to an adjacent agent  $A_j$  if, for each configuration  $R_i$  associated with  $A_i$  ( $R_i \in \bowtie_{R \in \Lambda'_i} R$ ), there exists a consistent configuration associated with  $A_j$ . A LJF is *directional interface-consistent* relative to a root agent if, for every two agents  $A_i$  and  $A_j$  where  $A_i$  is the parent of  $A_j$  relative to the root,  $A_i$  is interface-consistent relative to  $A_j$ .

The following two algorithms achieve directional interface-consistency in a LJF. The first below is used by agent  $A_i$  to update constraints in its linkage hosts based on constraint message from an adjacent agent  $A_j$ .

**Algorithm 5 (AbsorbInterfaceConstraint).** *When agent  $A_i$  performs AbsorbInterfaceConstraint relative to agent  $A_j$  with a set  $\Omega = \{R_X\}$ , where each  $R_X$  is a constraint over a linkage  $X$  with agent  $A_j$ ,  $A_i$  does the following:*

*for each linkage  $C$  with  $A_j$  with linkage host  $Q$  at  $A_i$ ,*  
*assign  $R_Q = R_Q \bowtie R_C$ , where  $R_C \in \Omega$ ;*  
*if  $R_Q = \emptyset$ , return false;*  
*return true;*

The second algorithm below recursively propagates constraint messages inwards along the hypertree. The agent executing the algorithm is referred to as  $A_0$  with local JT  $T_0$ . The execution is activated by a *caller* agent, who is either an adjacent agent, denoted by  $A_c$ , or the coordinator. Additional adjacent agents of  $A_0$  are denoted by  $A_1, \dots, A_m$ , if any.

**Algorithm 6 (CollectInterfaceConstraint).** *When caller calls  $A_0$  to CollectInterfaceConstraint, it does the following:*

```

1  for each agent  $A_i$  ( $i = 1, \dots, m$ ),
2    call CollectInterfaceConstraint on  $A_i$ ;
3    if  $A_i$  returns  $\emptyset$ , return  $\emptyset$ ;
4    receive  $\Omega_i = \{R_C\}$  where  $R_C$  is a constraint over a linkage  $C$  with  $A_i$ ;
5    perform AbsorbInterfaceConstraint relative to  $A_i$  with  $\Omega_i$ ;
6    if false is returned, return  $\emptyset$ ;
7  perform UnifyConstraint;
8  if false is returned, return  $\emptyset$ ;
9  if  $A_c$  is an adjacent agent,
10   initialize  $\Omega_c = \emptyset$ ;
11  for each linkage  $S$  with  $A_c$  of linkage host  $Q$  at  $A_0$ ,
12    compute  $R_S = \pi_S(R_Q)$ ;
13    if  $R_S = \emptyset$ , return  $\emptyset$ ;
14    else add  $R_S$  to  $\Omega_c$ ;
15  return  $\Omega_c$  to  $A_c$ ;
16 else return a special set  $\nabla$  to coordinator signifying successful completion;
```

Theorem 2 shows the consistency properties achieved by the above algorithm.

**Theorem 2.** *Let  $\mathcal{F} = (V, T, L, A')$  be the LJF representation of an MSCN populated by agents and CollectInterfaceConstraint is called on any agent  $A_0$ .*

*$\mathcal{F}$  is inconsistent iff  $A_0$  returns  $\emptyset$ . Otherwise,  $A_0$  returns  $\nabla$  and the following holds:*

1.  $\mathcal{F}$  is directional interface-consistent relative to  $A_0$ .
2. Each  $T_i$  is full arc-consistent.
3. Each linkage tree  $L_i$  is full arc-consistent.

The following algorithm generates a (partial) solution for a subdomain constrained by a partial solution over the interface with the calling agent.

**Algorithm 7 (GetLocalSolution).** *When agent  $A_0$  performs GetLocalSolution with  $\Omega = \{R_X\}$ , where each  $R_X$  is a singleton constraint (consisting of a single configuration) over a linkage  $X$  with agent  $A_c$ , it does the following:*

```

if  $\Omega = \emptyset$ , call DistributeSeparatorSolution in any cluster in  $T_0$ ;
else
  for each linkage  $C$  with  $A_c$  (whose host cluster is  $Q$ ),
    assign  $R_Q = R_Q \bowtie R_C$ , where  $R_C \in \Omega$ ;
  call DistributeSeparatorSolution in the host of any linkage with  $A_c$ ;
```

Note that after the assignment,  $R_Q$  is not necessarily a singleton. After `DistributeSeparatorSolution` is called, it is so. The following recursive algorithm propagates partial solutions over agent interfaces along the hypertree.

**Algorithm 8 (DistributeSolution).** *When caller calls  $A_0$  to `DistributeSolution`, it does the following:*

- 1 *if caller is an adjacent agent,*
- 2     *receive  $\Omega = \{R_X\}$  where each  $R_X$  is a singleton constraint over linkage  $X$  with caller;*
- 3     *perform `GetLocalSolution` with  $\Omega$ ;*
- 4 *else perform `GetLocalSolution` with  $\emptyset$ ;*
- 5 *for each agent  $A_i$  ( $i = 1, \dots, m$ ),*
- 6     *initialize  $\Omega' = \emptyset$ ;*
- 7     *for each linkage  $C$  with  $A_i$  (whose host cluster is  $Q$ ), add  $\pi_C(R_Q)$  to  $\Omega'$ ;*
- 8     *call `DistributeSolution` on  $A_i$  with  $\Omega'$ ;*

The following algorithm is executed by the system coordinator.

**Algorithm 9 (SolveDCSP).** *Choose an agent  $A^*$  arbitrarily. Call `CollectInterfaceConstraint` in  $A^*$ . If  $A^*$  returns  $\emptyset$ , return failure. Otherwise, call `DistributeSolution` in  $A^*$ .*

Theorem 3 below establishes that `SolveDCSP` is sound and complete.

**Theorem 3.** *Let  $\mathcal{F} = (V, T, L, A')$  be a LJF of an MSCN and `SolveDCSP` be executed. Then failure will be returned iff  $\mathcal{F}$  is inconsistent. Otherwise,  $R' = \bowtie_i (\bowtie_{Q \in T_i} R_Q)$  is a singleton such that  $R' \subseteq R$ , where  $R$  is the solution set of  $\mathcal{F}$ .*

Let  $n$  be the number of agents,  $t$  the maximum number of clusters in a local JT,  $q$  the maximum size of clusters, and  $k$  bind the space for variables in  $V$ . After `CollectInterfaceConstraint` completes, `SolveDCSP` is backtracking free. Hence, computation is dominated by `UnifyConstraint` during `CollectInterfaceConstraint`. `UnifyConstraint` has no more than twice the amount of computation of `CollectSeparatorConstraint`, whose complexity is  $O(t l^2)$  (Section 3), where  $l$  binds the size of relation in each cluster. Instead, we use a conservative complexity estimation,  $O(t k^{2q})$ , by replacing  $l$  with  $k^q$ . Therefore, the complexity of `SolveDCSP` is  $O(n t k^{2q})$ . When  $q$  is upper bounded, `SolveDCSP` is efficient. Note that  $q$  characterizes the density of an MSCN and is equivalent to the tree width of a centralized CN.

Another advantage of our method is that private variables in each agent and constraints over them are kept private during compilation and solution.

## 7 Example

We illustrate solution process for the example MSCN. Its compiled LJF is shown in Fig. 2. Initial constraints for clusters are listed in Table 1, where relations of the ‘same’ set of configurations are listed only once. For instance, relations over

**Table 1.** Initial constraints associated with clusters. A single line separates variables of relations with identical set of configurations which are enclosed by double lines.

$R_1$	0 0 2 1	$R_2$	f n p	1 0 2	2 1 0	$R_3$	0 0 1	0 2 2	1 2 0	2 1 1
d e g h	1 1 0 2	g h i	0 0 1	1 1 0	2 2 0	d e r	0 0 2	1 0 0	1 2 2	2 2 0
a b j k	1 1 2 0	b c m	0 0 2	1 1 2	2 2 1	a b r	0 1 1	1 0 2	2 0 0	2 2 1
c f t u	2 2 0 1	e r v	0 1 2	1 2 0		b c t	0 1 2	1 1 0	2 0 1	
0 0 1 2	2 2 1 0	b r s	0 2 1	2 0 1		c f n	0 2 1	1 1 2	2 1 0	

**Table 2.** Constraints as messages between clusters or newly assigned to clusters

$R_4$	g h	2 0	$R_5$	0 1 2	2 1 0	$R_6$	1 1	$R_7$	0 2 1	2 0 1	$R_8$	0 0 2
b r	0 1	2 1	b r s	0 2 1		a b	2 2	b c t	1 0 2	2 1 0	a b r	1 1 0
c t	0 2		e r v	1 0 2		c f		0 0 1	1 1 0	2 2 0	c f n	1 1 2
e r	1 0		f n p	1 2 0		d e		0 0 2	1 1 2	2 2 1	d e r	2 2 0
f n	1 2		g h i	2 0 1		0 0		0 1 2	1 2 0		0 0 1	2 2 1

clusters  $\{g, h, i\}$  and  $\{b, c, m\}$  are shown in the middle and will be referred to as  $R_2$  over  $\{g, h, i\}$  and  $R_2$  over  $\{b, c, m\}$ , respectively.

Suppose SolveDCSP is executed with  $A^* = A_0$ . Then, CollectInterfaceConstraint is called in  $A_0$ . In turn,  $A_0$  calls CollectInterfaceConstraint in  $A_1$ , which calls CollectInterfaceConstraint in  $A_2$  and  $A_3$ .

$A_3$  performs UnifyConstraint by calling CollectSeparatorConstraint in cluster, say,  $\{g, h, i\}$ , which in turn calls CollectSeparatorConstraint in cluster  $\{d, e, g, h\}$ . In response,  $\{d, e, g, h\}$  sends relation  $R_4$  (Table 2) over  $\{g, h\}$  to  $\{g, h, i\}$ , which causes modification of the constraint at  $\{g, h, i\}$  to  $R_5$  (Table 2).

Next,  $A_3$  calls DistributeSeparatorConstraint in  $\{g, h, i\}$ , which in turn calls DistributeSeparatorConstraint in  $\{d, e, g, h\}$  with  $R_4$  (Table 2). This results in no change in the constraint at  $\{d, e, g, h\}$ . UnifyConstraint at  $A_3$  returns with true.  $T_3$  is full arc-consistent with cluster constraints:  $R_1$  (Table 1) for  $\{d, e, g, h\}$  and  $R_5$  (Table 2) for  $\{g, h, i\}$ . Before completing CollectInterfaceConstraint,  $A_3$  sends  $A_1$  a message containing constraint  $R_6$  (Table 2) over linkage  $\{d, e\}$ .

At the same time,  $A_2$  also performs UnifyConstraint by calling CollectSeparatorConstraint in cluster, say,  $\{a, b, j, k\}$ , followed by calling DistributeSeparatorConstraint in  $\{a, b, j, k\}$ . During CollectSeparatorConstraint, the message from  $\{b, c, m\}$  to  $\{a, b, j, k\}$  is a universal relation over  $\{b\}$ , which causes no change in  $\{a, b, j, k\}$ . During DistributeSeparatorConstraint, the message from  $\{a, b, j, k\}$  to  $\{b, c, m\}$  is the same universal relation that causes no change in  $\{b, c, m\}$ . UnifyConstraint at  $A_2$  returns with true and  $T_2$  is full arc-consistent. Before completing CollectInterfaceConstraint,  $A_2$  sends  $A_1$  a message containing two constraints with one over each linkage. The constraint over  $\{a, b\}$  is  $R_6$  (Table 2) and that over  $\{b, c\}$  is universal.

After  $A_1$  receives the message from  $A_3$ , it calls AbsorbInterfaceConstraint, which causes the constraint at linkage host  $\{d, e, r\}$  to be modified into the relation  $R_8$  (Table 2). Similarly, after receiving the message from  $A_2$ ,  $A_1$  calls



AbsorbInterfaceConstraint. It modifies the constraint at linkage host  $\{a, b, r\}$  into the relation  $R_8$  (Table 2) but constraint at linkage host  $\{b, c, t\}$  remains as  $R_3$  (Table 1).

Subsequently,  $A_1$  performs UnifyConstraint by calling CollectSeparatorConstraint in cluster, say,  $\{a, b, r\}$ , followed by calling DistributeSeparatorConstraint. During CollectSeparatorConstraint, the message sent from  $\{e, r, v\}$  to  $\{d, e, r\}$  is a universal relation over  $\{e, r\}$  and hence causes no change to the constraint at  $\{d, e, r\}$ . The message sent from  $\{d, e, r\}$  to  $\{a, b, r\}$  is a universal relation over  $\{r\}$  and hence causes no change to the constraint at  $\{a, b, r\}$ . The message from  $\{b, r, s\}$  to  $\{a, b, r\}$  is a universal relation over  $\{b, r\}$  and causes no change to the constraint at  $\{a, b, r\}$ . The message from  $\{c, f, t, u\}$  to  $\{b, c, t\}$  is  $R_4$  (Table 2) over  $\{c, t\}$  and changes the constraint at  $\{b, c, t\}$  to  $R_7$  (Table 2). The message from  $\{b, c, t\}$  to  $\{a, b, r\}$  is universal over  $\{b\}$  and causes no change to constraint at  $\{a, b, r\}$ .

During DistributeSeparatorConstraint, the message from  $\{a, b, r\}$  to  $\{d, e, r\}$  is a universal relation over  $\{r\}$  and causes no change to the constraint at  $\{d, e, r\}$ . The message from  $\{d, e, r\}$  to  $\{e, r, v\}$  is  $R_4$  (Table 2) over  $\{e, r\}$  and it modifies the constraint at  $\{e, r, v\}$  to  $R_5$  (Table 2). The message from  $\{a, b, r\}$  to  $\{b, r, s\}$  is  $R_4$  (Table 2) over  $\{b, r\}$  and modifies the constraint at  $\{b, r, s\}$  to  $R_5$  (Table 2). The message from  $\{a, b, r\}$  to  $\{b, c, t\}$  is a universal relation over  $\{b\}$  and causes no change to the constraint at  $\{b, c, t\}$ . The message from  $\{b, c, t\}$  to  $\{c, f, t, u\}$  is  $R_4$  (Table 2) over  $\{c, t\}$  and causes no change to the constraint at  $\{c, f, t, u\}$ . UnifyConstraint at  $A_1$  returns with true.  $T_1$  is full arc-consistent with the following cluster constraints:  $R_1$  (Table 1) for  $\{c, f, t, u\}$ ,  $R_7$  (Table 2) for  $\{b, c, t\}$ ,  $R_8$  (Table 2) for  $\{d, e, r\}$  and  $\{a, b, r\}$ ,  $R_5$  (Table 2) for  $\{e, r, v\}$  and  $\{b, r, s\}$ . Before completing CollectInterfaceConstraint,  $A_1$  sends  $A_0$  a message containing constraint  $R_6$  (Table 2) over linkage  $\{c, f\}$ .

After  $A_0$  receives the message, it calls AbsorbInterfaceConstraint which replaces the constraint at linkage host  $\{c, f, n\}$  by  $R_8$  (Table 2). Afterwards,  $A_0$  performs UnifyConstraint by calling CollectSeparatorConstraint in cluster, say,  $\{f, n, p\}$ , followed by calling DistributeSeparatorConstraint. During CollectSeparatorConstraint, the message from  $\{c, f, n\}$  to  $\{f, n, p\}$  is  $R_4$  (Table 2) over  $\{f, n\}$ . It modifies the constraint at  $\{f, n, p\}$  into  $R_5$  (Table 2). During DistributeSeparatorConstraint, the message from  $\{f, n, p\}$  to  $\{c, f, n\}$  is  $R_4$  (Table 2) over  $\{f, n\}$  and has no effect at  $\{c, f, n\}$ . UnifyConstraint at  $A_0$  returns with true.  $T_0$  is full arc-consistent with the following cluster constraints:  $R_8$  (Table 2) for  $\{c, f, n\}$  and  $R_5$  (Table 2) for  $\{f, n, p\}$ . As the result,  $A_0$  terminates CollectInterfaceConstraint and returns  $\nabla$ .

Subsequently,  $A_0$  is called to DistributeSolution. It runs GetLocalSolution by first calling DistributeSeparatorSolution at, say,  $\{f, n, p\}$ . This produces the partial solution  $R_{11}$  for  $\{f, n, p\}$  first and then  $R_{10}$  (Table 3) for  $\{c, f, n\}$  at  $T_0$ .

Next,  $A_0$  calls  $A_1$  to DistributeSolution with the message containing the relation  $R_{12}$  (Table 3) over  $\{c, f\}$ . In response,  $A_1$  modifies its constraint in linkage host  $\{c, f, t, u\}$  to  $R_9$ . It then calls DistributeSeparatorSolution in the host  $\{c, f, t, u\}$ . The resultant partial solution at each cluster of  $T_1$  are as follows:  $R_{13}$

**Table 3.** Relations generated during DistributeSolution

$R_9$	$R_{10}$	c f n	$R_{11}$	2 1 0	$R_{12}$	d e	$R_{13}$	2 2 1 0	$R_{14}$
c f t u	a b r	d e r	b r s		a b	2 2	a b j k		g h i
2 2 0 1	b c m	2 2 1	e r v		b c		c f t u		1 0 2
2 2 1 0	b c t		f n p		c f		d e g h		

over  $\{c, f, t, u\}$ ,  $R_{10}$  over  $\{b, c, t\}$ ,  $\{a, b, r\}$ ,  $R_{11}$  over  $\{b, r, s\}$ ,  $R_{10}$  over  $\{d, e, r\}$ , and  $R_{11}$  over  $\{e, r, v\}$ .

After that,  $A_1$  calls  $A_2$  to DistributeSolution with the message containing relations  $R_{12}$  over  $\{a, b\}$  and  $\{b, c\}$ . In response,  $A_2$  generates partial solutions  $R_{13}$  (Table 3) over  $\{a, b, j, k\}$  and  $R_{10}$  over  $\{b, c, m\}$  at  $T_2$ .

Similarly,  $A_1$  calls  $A_3$  to DistributeSolution with the message containing relation  $R_{12}$  over  $\{d, e\}$ . In response,  $A_3$  generates partial solutions  $R_{13}$  over  $\{d, e, g, h\}$  and  $R_{14}$  over  $\{g, h, i\}$  at  $T_3$ . SolveDCSP now terminates successfully and the natural join of the above partial solutions in all agents is the solution.

## 8 Conclusion

In this contribution, we proposed a representation of DCSPs as MSCNs, extended techniques for MSBNs [10] to compilation of MSCNs into runtime LJFs, and presented the first algorithm suite that solves efficiently DCSPs of bounded primal graph density. The algorithm suite is shown to be sound and complete. Therefore, we have shown that MSCNs form a tractable class of DCSPs. Experimental study on distributed scheduling is underway.

## Acknowledgement

Financial support to the first author through Disvoery Grant from NSERC, Canada is acknowledged.

## References

1. R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.
2. R. Dechter and J. Pearl. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34:1–38, 1988.
3. R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366, 1989.
4. C.L. Mason and R.R. Johnson. DATMS: a framework for distributed assumption based reasoning. In L. Gasser and M.N. Huhns, editors, *Distributed Artificial Intelligence II*, pages 293–317. Pitman, 1989.
5. P. Meseguer and M. Jimenez. Distributed forward checking. In *Proc. CP'2000 Distributed Constraint Satisfaction Workshop*, 2000.
6. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

7. M.C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with aggregations. In *Proc. AAAI'2000*, pages 917–922, 2000.
8. M.C. Silaghi, D. Sam-Haroud, and B. Faltings. ABT with asynchronous reordering. In *Proc. Inter. Conf. on Intelligent Agent Technology*, pages 54–63, 2001.
9. M.C. Silaghi, D. Sam-Haroud, and B. Faltings. Consistency maintenance for ABT. In *Principles and Practice of Constraint Programming - Proc. CP2001, LNCS 2239*, pages 271–285. Springer-Verlag, Berlin, 2001.
10. Y. Xiang. *Probabilistic Reasoning in Multi-Agent Systems: A Graphical Models Approach*. Cambridge University Press, Cambridge, UK, 2002.
11. M. Yokoo. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In *Principles and Practice of Constraint Programming - CP95, Lecture Notes in Computer Science, Vol.976*, pages 88–102. Springer-Verlag, 1995.
12. M. Yokoo. *Distributed Constraint Satisfaction*. Springer, 2001.
13. M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara. Distributed constraint satisfaction for formalizing distributed problem solving. In *Proc. 12th IEEE Inter. Conf. on Distributed Computing Systems*, pages 614–621, 1992.
14. M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proc. 2nd Inter. Conf. on Multi-Agent Systems*, pages 401–408, 1996.

# A Clustering Algorithm Based on Adaptive Subcluster Merging

Jiani Hu, Weihong Deng, and Jun Guo

Beijing University of Posts and Telecommunications, 100876, Beijing, China  
{cughu, cvpr\_dwh}@126.com, guojun@bupt.edu.cn

**Abstract.** This paper proposes an adaptive subcluster merging (ASM) based clustering algorithm. The ASM algorithm has two stages: subcluster partition and subcluster merging. Specifically, it first applies local expanding with variance constraint to partition subclusters with uniform granularity, and then it adaptively merges the subclusters into clusters with the notion of density. Through these two stages, ASM algorithm can identify clusters of heterogeneous structures. The feasibility of the algorithm has been successfully tested on both synthetic and real-world data sets. Comparative experimental studies of various clustering algorithms are also performed. The results demonstrate that the proposed algorithm performs better than K-means, complete-link hierarchical, density-based and maximum variance algorithms.

## 1 Introduction

Clustering analysis is an unsupervised learning technique used to discover group structure of a data set. The goal of clustering is to group a set of patterns, points, or objects into meaningful subsets whose in-class members are “similar” in some sense and whose cross-class members are “dissimilar”. The clustering problem has been extensively studied in many scientific disciplines and a variety of different algorithms have been developed [2,7,6,10,11].

Generally speaking, clustering algorithms can be categorized into hierarchical and partitional algorithms [4]. Hierarchical algorithms deliver a hierarchy of possible clusterings, while partitional clustering algorithms divide the data up into a number of subsets. In partitional clustering analysis, many algorithms, such as the K-means [4] and Gaussian Mixture Model [9], assume the number of clusters to be known a priori. Other algorithms partition clusters by minimizing the cluster scatter with a constraint on certain measures, such as cluster variance [11]. Since the real data clusters are usually heterogeneous, the structure of a data set probably can not be found by simply imposing a variance constraint.

In this paper, we propose an adaptive subcluster merging (ASM) based clustering algorithm, which is capable of discovering the heterogeneous structures of the data set. Specifically, the ASM algorithm has two stages: subcluster partition and subcluster merging. The subcluster partition stage applies local expanding, i.e. expanding the subcluster by its nearest neighbor under the condition that the variance of the expanded subcluster is less than a threshold. After this stage,

all patterns in a data set will be partitioned into a number of subclusters of similar granularity. The second stage is to merge the subclusters according to the density of subclusters' borders. By these two stages, ASM algorithm overcomes the homogenous limitation of the resultant clusters of MVC [11]. It also avoids the shortcoming of density consistency in DBSCAN algorithm [10]. Experimental results on synthetic data and UCI data [8] show our algorithm performs better than other popular clustering algorithms, e.g. K-means [4], complete-link hierarchical [4], density-based [10], and MVC algorithm, in terms of Fowlkes and Mallows index [5,3].

The rest paper is organized as follows. Section 2 describes the proposed algorithm. Detailed experimental evaluations among our algorithm and other famous clustering algorithms are shown in section 3. Finally, section 4 provides some concluding remarks.

## 2 The Adaptive Subcluster Merging Algorithm

### 2.1 Subcluster Partition

“Subcluster” is a set of homogeneous data whose variance is within a constraint. It is anticipated to be a subset of a real cluster. In order to partition the subclusters, a reduced maximum variance cluster algorithm [11] is applied. The key notion of subcluster partition is to gradually expand a hypothesized subcluster by its nearest neighbor under the condition that variance of current subcluster is below the constraint.

A subcluster begins with a “unlabeled” pattern in the data set. Then the subcluster is expanded by its nearest “unlabeled” pattern and the expanding process is terminated when the variance of the subcluster becomes larger than a constraint. After the accomplishment of partitioning a subcluster, a new subcluster is started to be partitioned in the same way. Subclusters are partitioned one by one until there is no “unlabeled” pattern left in the data set. The main issue of subcluster partition lies in searching for the nearest neighbor of a subcluster. Here a convenient method is utilized to solve this issue. It is evident that the neighbor of a subcluster is also the neighbor of the subcluster's border. Once the border is identified, the nearest neighbor of the subcluster is easy to identify. A subcluster's border consists of those samples that are the furthest cluster mates of the samples in the subcluster. Let  $B_i$  stand for the border of the  $i$ th subcluster  $C_i$ . Accordingly, the  $k$ th order border of the  $i$ th subcluster, called  $B_i(k)$ , can be expressed as follows:

$$B_i(k) = \bigcup_{x \in C_i} I(x, k, C_i) \quad (1)$$

where  $I(x, k, C_i)$  is the set of  $k$  furthest samples of  $x$  in  $C_i$ . In other words, the  $k$  furthest neighbors of  $x$  in subcluster  $C_i$  are:

$$I(x, k, C_i) = \begin{cases} f(x, C_i) \cup I(x, k-1, C_i - f(x, C_i)), & \text{if } k > 0 \\ \emptyset, & \text{if } k = 0 \end{cases} \quad (2)$$

where  $f(x, C_i)$  is the furthest neighbor of  $x$  in  $C_i$  and

$$f(x, C_i) = \arg \max_{y \in C_i} \|y - x\|^2. \quad (3)$$

The  $i$ th subcluster's border  $B_i$  can be found according to the above process. Thus, the nearest foreign neighbor of the subcluster  $C_i$ , which is also the nearest foreign neighbor of the border  $B_i$ , is defined as follows:

$$NN(C_i) = NN(B_i) = \arg \min_{y \notin C_i} \|y - x\|^2, \text{ where } x \in B_i. \quad (4)$$

From Equation(4), it can be observed that the nearest neighbor of the subcluster is also the expanding pattern which makes the minimal change to the variance of current subcluster.

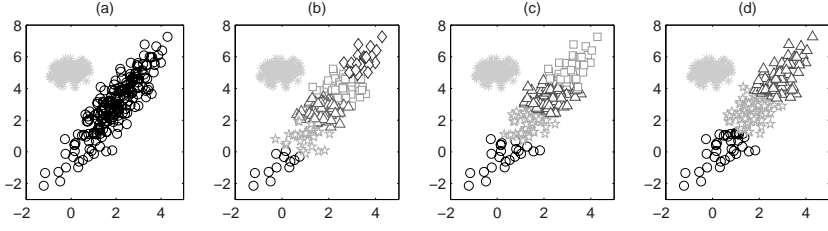
Obviously, the variance of a cluster is increasing step by step along with the local expanding process. If there is no constraints, the  $n$  patterns in the whole set  $\mathcal{D} = \{x_1, \dots, x_n\}$  will form a single subcluster. Here we propose a variance constraint( $T$ ) to all subclusters, that is  $\sigma_i^2 = \frac{\sum_{x \in \mathcal{D}_i} \|x - m_i\|^2}{n_i} \leq T$ , where  $n_i$  is the number of patterns in the  $i$ 's subcluster and  $m_i = \frac{1}{n_i} \sum_{x \in \mathcal{D}_i} x$  is the mean vector of the  $n_i$  patterns. The variance of a subcluster is computed after each expanding step. If it is higher than  $T$ , the subcluster will stop expanding. Though subcluster partition stage, the whole set  $\mathcal{D}$  may be split into a number of subclusters with same granularity.

## 2.2 Subcluster Merging

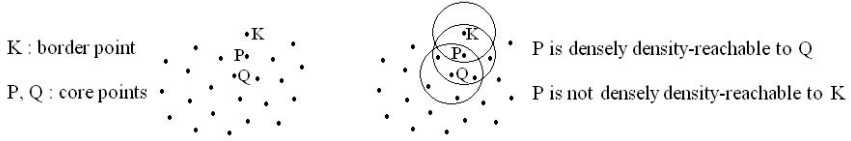
Apparently, resulting subclusters of the reduced MVC algorithm turn to be similar hyperspherical shapes [11]. However, clusters of real data sets may have different distributions, sizes, and shapes. Thus, this kind of solution might not be able to reveal real data structures. Let's take an example for explanation. Fig.1(a) shows a synthetic data set, which consists of two Gaussian distributed clusters signed with gray "\*" and dark "o". Varying the variance constraint  $T$  from 0.8 to 1.5, we can obtain the resulting subclusters proposed in Fig.1(b), (c), and (d) with different colors and signs. None of the resultant partitions can identify the real partition of the data set.

In order to avoid the homogenous limitation of MVC and discover heterogeneous structure of data, we introduce density-based subcluster merging stage. The density notion is that the density of the inner part of a cluster center is higher than that of the cluster border. With this notion, we can investigate the density of borders between two subclusters in order to determine whether to merge them.

Let's explicate the definition of density. The density of a point  $q$ , denoted as  $\rho(q, \varepsilon)$ , is described as the number of samples in set  $A$ , where  $A = \{p \mid \text{distance}(q, p) \leq \varepsilon\}$ . Here we propose the notion of *densely density-reachable*. If 1)  $\text{distance}(q, p) \leq \varepsilon$ , and 2)  $\rho(p, \varepsilon) \geq \rho_m$  ( $\rho_m$  is a certain density threshold), we regard point  $q$  is *densely density-reachable* to point  $p$ . *Densely density-reachable*



**Fig. 1.** The original data set and its clustering results with the variance constraint  $T=0.8, 1$ , and  $1.5$  respectively



**Fig. 2.** Concept of densely density-reachable

is symmetric for core point pairs. However, it is not symmetric if a border point is involved. For example core points, as point P and Q in Fig.2, are densely density-reachable to each other (with respect to  $\rho_m = 4$ ) while the core point P is not densely density-reachable to border point K. Thus, we can take some border point pairs from two different subclusters, whose distances are below  $\varepsilon$ , into consideration. If two border points of every pair are densely density-reachable to each other, they must be core points of a cluster. As a result, we merge these two subclusters.

The properties of every subclusters is utilized in order to determine the density parameter  $\varepsilon$  and  $\rho_m$ . Suppose average distances between every border point and its nearest neighbor in the  $i$ th subcluster is regarded as  $\varepsilon(i)$ . That is

$$\varepsilon(i) = \frac{\sum_{p \in B_i} \text{distance}(p, nn(p))}{k}, \quad (5)$$

where  $nn(p)$  is the nearest neighbor of  $p$ , i.e.  $nn(p) = \arg \min_{y \in C_i} \|y - p\|^2$ , and  $k$  is the total number of points in border  $B_i$ . And  $\rho_m(i)$  is regarded as the average density of border points with respect to the  $\varepsilon(i)$ ,

$$\rho_m(i) = \frac{\sum_{p \in B_i} \rho(p, \varepsilon(i))}{k}. \quad (6)$$

The merging rule of two subcluster ( $i$  and  $j$ ) can be detailed as follows. The density parameters are set by  $\varepsilon = \frac{\varepsilon(i) + \varepsilon(j)}{2}$  and  $\rho_m = \max(\rho_m(i), \rho_m(j))$ . If there exist 1) a border point pair set  $S = \{(x, y) \mid x \in B_i, y \in B_j, \text{and } \text{distance}(x, y) \leq \varepsilon\}$ ; and 2) all border point pairs satisfy  $\rho(x, \varepsilon) \geq \rho_m$  and  $\rho(y, \varepsilon) \geq \rho_m$ , i.e. they are all densely density-reachable border pairs; we merge the  $i$ th and  $j$ th subcluster.

## 2.3 Clustering Algorithm

The proposed two-stage algorithm has only one parameter, i.e. the variance constraint. The detailed description of the algorithm is proposed as follows.

---

### ASM Algorithm

**Input:** Data set  $\mathcal{D} \in \mathbb{R}^{n \times d}$  (all the  $n$  samples in  $d$ -dimension are marked as “unlabeled”), variance constraint  $T$

**Output:** Set of  $m$  clusters  $C_i$ , where  $\bigcup_{i=1}^m C_i = \mathcal{D}$

---

#### {Subcluster partition stage}

$i = 0$ ; % each cluster is given an identifier  $i$

$index = \text{randomPermutation}(1...n)$ ;

**for**  $j = 1$  **to**  $n$  **do**

$a = index(j)$ ;

**if**  $x_a$  is marked as “unlabeled” **then**

        mark  $x_a$  as “labeled”;

$i = i + 1$ ;

$C_i = \{x_a\}$ ;

$\sigma^2 = 0$ ;

**while**  $\sigma^2 < T$

            find the border  $B_i(k)$  of the cluster  $C_i$ ; %  $k$  is set to 5

            find the nearest neighbor of  $B_i$  in “unlabeled” patterns;

**if**  $NN(B_i) \neq \emptyset$

$C_i = C_i \cup NN(B_i)$ ;

                mark  $NN(B_i)$  as “labeled”;

                recompute the variance  $\sigma^2$  of the current cluster  $C_i$ ;

**else**

**break**;

**end while**;

**end if**;

**end for**;

#### {Subcluster merging stage}

**for**  $j = 1$  **to**  $i$  **do**

**for**  $k = j + 1$  **to**  $i$  **do**

        compute density merging parameters, i.e.  $\varepsilon = \frac{\varepsilon(j) + \varepsilon(k)}{2}$  and  $\rho_m = \max(\rho_m(j), \rho_m(k))$ ;

        find the pattern pair set in the  $j$ th and  $k$ th borders,

        i.e.  $S = \{(x, y) \mid x \in B_j, y \in B_k, \text{and } distance(x, y) \leq \varepsilon\}$ ;

**if**  $S \not\subset \emptyset$  **and** two patterns in each pair of  $S$  are all densely density-reachable to each other,

            save the cluster identifiers  $j$  and  $k$ , which can be merged;

**end if**;

**end for**;

**end for**;

    update cluster identifier  $i$ ;

---

The space complexity of ASM algorithm is  $O(n^2)$ . This is because a similarity matrix of size  $n \times n$  has to be stored, where  $n$  denotes the number of samples in the dataset. Its time complexity is  $O(n^2)$ . Typically,  $n$  is fixed in advance and so the algorithm has linear computational complexity in the size of the data set.

## 3 Experiments and Analysis

### 3.1 Data Sets and Evaluation Measure

In order to assess the feasibility and performance of our ASM algorithm, experiments are done on both synthetic data shown in Fig.1(a) and some real-world



**Table 1.** Summary of datasets used in the experiments

Data set	Num. of samples	Num. of clusters	Num. of features
Synthetic data	325	2	2
Wisconsin Breast Cancer	699	2	9
Iris	150	3	4
Glass	214	6	9
Handwritten numerals	2000	10	64

data sets, available in the UCI Machine Learning Repository [8]. The details of the datasets are listed in Table 1.

The quality of a clustering solution is evaluated using the *Fowlkes and Mallows index* [5,3], which measures the similarity of resulting clusters with real clusters of a data set. Consider  $C = \{C_1, \dots, C_m\}$  is a clustering structure of a data set  $\mathcal{D}$ , and  $P = \{P_1, \dots, P_s\}$  is the actual partition of the data. The state of each pair of samples  $(x_q, x_k)$  pertains to one of the following four states: 1) SS: if both samples belong to the same cluster of the resulting cluster  $C$  and to the same group of partition  $P$ ; 2) SD: if samples belong to the same cluster of  $C$  and to different groups of  $P$ ; 3) DS: if samples belong to different clusters of  $C$  and to the same group of  $P$ ; 4) DD: if both samples belong to different clusters of  $C$  and to different groups of  $P$ . Assuming that  $a, b, c$  and  $d$  are the number of SS, SD, DS, DD pairs respectively. The *Fowlkes and Mallows index* is then defined as:

$$FM = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}} \tag{7}$$

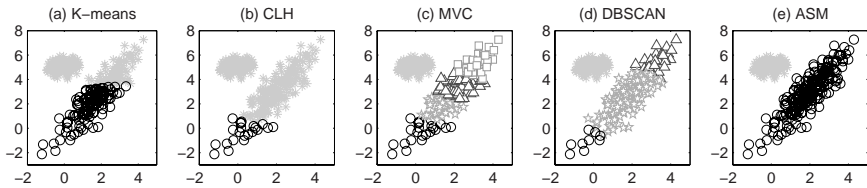
Apparently, a higher value of  $FM$  means the resultant partition  $C$  is more similar to the real partition  $P$ . A perfect clustering solution leads the value of  $FM$  to 1.

**3.2 Experimental Results**

Our experiments are started with the synthetic data shown in Fig.1(a) consisting of two Gaussian clusters. We compare the ASM clustering algorithm with K-means, complete-link hierarchial (CLH)[4], maximum variance cluster (MVC) [11], and density-based (DBSCAN) algorithms in terms of the Fowlkes and Mallows (FM) index . For fair comparisons, the optimal parameter of each algorithm is reported in Table 2. As K-means, MVC, DBSCAN, and ASM are sensitive to the randomly selected initial patterns, each algorithm is run 10 times. The average FM index and standard deviation (std) of all algorithms are reported in Table 2 while the graphic resultant partitions are shown in Fig.3. The results show that K-means algorithm is the most sensitive to the randomly initialized patterns. MVC tends to partition clusters into hyperspherical shapes. DBSCAN can detect the high density regions but fails to identify the low density borders. This observation of DBSCAN is the same with [12]. ASM is the only algorithm

**Table 2.** Statistical results of different algorithms on Synthetic data

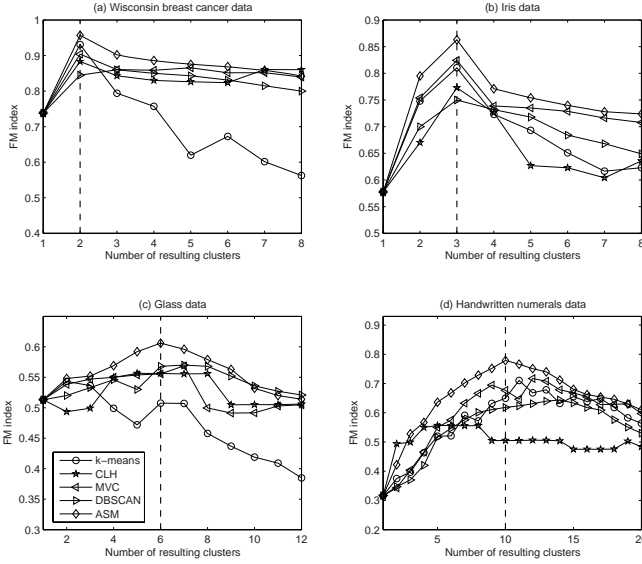
Algorithm	Parameter	FM $\pm$ std	Num. of result clusters
K-means	k=2	$0.650 \pm 0.014$	2
CLH	t=8	$0.667 \pm 0$	2
MVC	$\sigma^2 = 1.2$	$0.699 \pm 0.002$	5
DBSCAN	$MinPts = 4, Eps = 0.18$	$0.812 \pm 0.003$	4
ASM	$T \in [0.8, 1.5]$	$1 \pm 0$	2


**Fig. 3.** Clustering results of five algorithms on synthetic data

which can not only obtain the highest FM index but also identify the real structure of the data set. Furthermore, varying the parameter  $T$  from 0.8 to 1.5, the ASM algorithm can always obtain perfect performance. That is the reason the algorithm is called “adaptive”.

The following experiments compare ASM algorithm with other four algorithms (K-means, CLH, MVC, and DBSCAN) using four real-world data sets in Table 1. We vary the parameters of different algorithms to get different number of resulting clusters, and compare their FM index under the same number of resulting clusters. Apparently, for the same number of resulting clusters a higher FM index indicates the better quality of the clustering solution. Comparative performance of the five algorithms on four different data sets are presented in Fig.4. The main observations from these experiments are:

1. The performances of K-means algorithm fluctuate dramatically with the changing of result cluster numbers as shown in Fig.4(a) and (c).
2. In general, the performances of complete-link hierarchical method are not satisfying. It manipulates a dissimilarity matrix between patterns, imposes a hierarchical structure on data. This property indicate the CLH algorithm only considers local neighbors in each step, the global shape and size of clusters are always ignored [1]. Thus, unbalanced clusters are not adequately handled by this method.
3. The performance of MVC and DBSCAN are similar in the four experiments. A common point of these two algorithms is that their peak values are not obtained when the resulting cluster number is equal to real class number.
4. The FM indexes of our proposed method are generally higher than other algorithms under the same number of resulting cluster. In addition, the highest



**Fig. 4.** Comparative performance of the five algorithms on four different data sets. (the vertical dashed line indicates the real cluster number)

FM indexes are obtained when the resulting cluster number equal to the real cluster number. A problem to be noted is that when the number of resulting clusters get too large the superiority of ASM is not evident. It is because that the subclusters are so many that merging of them accurately is difficult.

### 3.3 Discussion

In this section, we discuss the effect of the variance parameter of ASM algorithm. As the demonstration in section 2, the parameter dominates the granularity of the subclusters. A too small parameter will produce the subclusters which has only one sample in, thus the ASM algorithm will be similar to DBSCAN which merges the samples in the data set one by one. On the other hand, a too large parameter will induce all the samples be in a single “subcluster”, therefore merging process would be omitted. As long as the parameter is neither larger than the minimal variance of the real clusters nor too small, the ASM algorithm can adaptively merge the subclusters. Experimental results in Table 2 demonstrate that a wide range of  $T$  can obtain expecting results. It is one of superiority of our algorithm that a wide range of parameters are suitable to obtain its best performance.

## 4 Conclusions

This paper proposes an adaptive subcluster merging based clustering algorithm, which can be separated into subcluster partition and subcluster merging stages.

Experiments have proven that the ASM algorithm is generally superior to K-means, CLH, MVC, and DBSCAN algorithms. It is capable of identifying heterogeneous cluster structures, and also flexible on parameter selection. A drawback of the AGM algorithm may be lie in its computational complexity. Our next goal is to utilize other clustering algorithms to the subcluster partition stage and optimize its computational efficiency.

## Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No.60475007, 60675001), Key Project of Foundation of Ministry of Education of China (Grant No.02029), and Cross-Century Talents Foundation.

## References

1. Fred, A.L.N, Leitaio, J.M.N.: A new cluster isolation criterion based on dissimilarity increments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25** (2003) 944–958
2. Guha, S., Rastogi, R., and Shim, K.: CURE: an efficient clustering algorithm for large databases. In *Proc. of 1998 ACM-SIGMOD International Conference on Management of Data* (1998) 73–84
3. Halkidi, M., Batistakis, Y., and Vazirgiannis, M.: Cluster validity methods: part I. *ACM SIGMOD Record*, **55** (2002) 311–331
4. Jain, A.K., Murty, M.N., and Flynn, P.J.: Data clustering: a review. *ACM Computing Surveys* **31** (1999) 264–323
5. Jain, A.K., Dubes, R.C.: *Algorithms for clustering data*. Prentice Hall (1988)
6. Karypis, G., Han, E., and Kumar, V.: Chameleon: a hierarchical clustering algorithm using dynamic modeling. *IEEE Computer* **32** (1999)
7. McLachlan, G.J., and Basford, K.E.: *Mixture models: inference and applications to clustering*. New York: Marcel Dekker (1988)
8. Newman, C.B.D.J., Hettich, S., and Merz, C.: *UCI repository of machine learning database*. (1998)
9. Roberts, S., Husmeier, D., Rezek, I., and Penny, W.: Bayesian approaches to gaussian mixture modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20** (1998) 1133–1142
10. Sander, J., Ester, M., Kriegel, H.P., and Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd Int. Conf. on Knowledge Discovery and Data Mining* (1996) 226–231
11. Veenman, C.J., Reinders, M.J.T., and Backer, E.: A maximum variance cluster algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24** (2002) 1273–1280
12. Yip, A.M., Ding, C., and Chan, T.F.: Dynamic cluster formation using level set methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28** (2006) 877–889

# Efficient Algorithms for Video Association Mining

B. SivaSelvan<sup>1</sup> and N.P. Gopalan<sup>2</sup>

<sup>1</sup> Department of Computer Science & Engineering  
National Institute of Technology Tiruchirapalli, India 620 015  
[siva.selvan@gmail.com](mailto:siva.selvan@gmail.com)

<sup>2</sup> Department of Computer Applications,  
National Institute of Technology Tiruchirapalli, India 620 015  
[gopalan@nitt.edu](mailto:gopalan@nitt.edu)

**Abstract.** Video Association Mining(VAM) is the process of discovering associations in a given video. Two key phases of VAM are (i) Transformation and (ii) Frequent Temporal Pattern Mining. The transformation phase converts the original input video to an alternate transactional format, namely a cluster sequence. Frequent temporal pattern mining phase concerns the generation of patterns subject to the temporal distance and support thresholds. The paper addresses the issue of frequent temporal pattern mining and studies algorithms for the same. The existing Apriori based algorithm is compared with three other approaches highlighting the case specific situations suited by each.

## 1 Introduction

Data Mining is the nontrivial process of extraction of previously unknown and potentially useful information from large databases. A few of the existing data mining techniques are Classification, Clustering, Association Rule Mining, Prediction, Outlier Analysis, etc [1]. Association Rule Mining is the process of generating associations and hence identifying interesting relationships among database items. It finds application in Market Basket Analysis to identify frequently purchased items and establish relationships among them [1,2]. Data Classification is a supervised mining technique that generates class information or labels for input test data. Clustering, an unsupervised technique groups input data based on the principle of maximising intra cluster and minimising inter cluster similarity. These data mining technique have been well explored in the context of conventional and transactional databases.

Modern day information technology and internet revolution has resulted in enormous amounts of multimedia datasets such as images, videos, etc. With large volumes of multimedia data available, Multimedia Data Mining (MDM) is an emerging trend in knowledge extraction research. It deals with application of data mining techniques to discover high level multimedia information [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22]. MDM differs from its conventional counterpart due to the presence of data specific properties in multimedia data.

Video data is governed by temporal properties while images are bound by spatial properties. Conventional data mining algorithms lacking these properties are not suited to multimedia data. MDM concentrates on adapting existing or evolving new data mining techniques to extract knowledge from multimedia data sets.

Data Mining techniques such as Classification, Clustering, Association Mining, Prediction find immediate application in the domain of multimedia data. Classification and Clustering are used for class determination or grouping of video and image inputs [10, 11, 16]. Association Mining is employed to establish relationships among the constituents of the multimedia data [10, 13, 14, 16, 17]. Associations mined from images and videos are also employed in classification to determine the overall nature of the video or image data [9, 11, 12, 18]. Reference [15] discusses various knowledge types that can be mined from video data. Amongst the various MDM techniques, Video Association Mining is a relatively new and emerging research trend and forms the focus of this paper.

The rest of the paper is organized as follows: Section 2 discusses Video Association Mining and its applications in greater detail. Frequent Temporal Pattern(FTP) mining, an important phase of VAM is explained in Section 3. The proposed algorithms for FTP mining are presented in Section 4. Section 5 performs a detailed complexity analysis of the proposed approaches in relation to Apriori based one and Section 6 presents the experimental results. Section 7 summarizes the results of our study and concludes with future research directions.

## 2 Video Association Mining

Video Association Mining is the process of discovering associations in a given video. Associations can be established between various objects in a frame that are representative of a scene. Also associations can be established at higher levels of abstraction such as relationships between director, movie type, etc. The generated associations could also be used to predict futuristic events based on the occurrence of a certain sequence of events frequently. Reference [15] discusses the different types of knowledge that can be mined from cinema and its applications. Generated associations can also be employed in video classification to determine the overall nature of the movie such as being romantic, comic, etc. Video associations are also employed in summarization by including the most frequent patterns in the summary [9, 12, 18].

References [9, 12, 18] discuss a technique of generating video associations by transforming the original video input data into an equivalent transactional data format. This is done by grouping the various shots of the original video into different clusters, each of which consists of visually similar shots. A shot cluster sequence consisting of cluster information of each shot arranged by its temporal order is constructed. The problem of mining video associations thus gets reduced to that of mining associations from cluster sequences. Two types of associations identified in [9, 12] are Intra and Inter associations. Intra associations are those in which all items involved in the association are the same. This could be a

result of scenes composed of visually similar shots of the same object taken from different viewpoints. Inter associations are those which consists of items of different types, which are scenes that consist of visually distinct shots of different objects.

Video mining differs from its other multimedia counterparts in the presence of temporal properties in the input data. References [9,12,18] incorporate the temporal aspect in the video association mining process via two parameters namely temporal support and distance thresholds. Temporal distance(TD) between two items or shots is the number of shots between them. The temporal distance of the pattern AB in the input sequence ACEB is 2. The support measure based on temporal distance is referred to as temporal support. It is the number of times the association is shown sequentially in the input video, subject to the temporal distance threshold value. For example in the input sequence ABABACABC, the temporal support of pattern ABC for  $TD=\infty$  is 2 and  $TD=0$  is 1. A temporal distance threshold value of  $\infty$  denotes infinite distance between the various possible patterns or shot clusters.

The existing video association mining technique employs two phases namely (i) Transformation and (ii) Mining. The transformation phase converts the original video data into an alternate transactional data format using clustering. By the end of the transformation phase, the problem of VAM gets reduced to mining frequent patterns from the transformed cluster sequence, subject to the temporal distance and support factors. This process of frequent pattern mining subject to the temporal factors is referred as Frequent Temporal Pattern (FTP) mining and is discussed in greater detail in the following section.

### 3 Frequent Temporal Pattern (FTP) Mining

FTP mining though similar to the frequent set mining phase of conventional association rule mining differs from its transactional counterparts in the temporal support and distance factors. Reference [8] provides a detailed survey of algorithms for frequent set mining in conventional data domain. The existing video association mining technique employs Apriori algorithm [2] in the process of FTP mining. Apriori, the first algorithm for frequent set mining is based on a level wise principle and the anti-monotone property of set theory that “Every subset of a frequent set is also frequent”.

Apriori based FTP mining algorithm [9,12,18] first constructs frequent patterns (item-sets) of length 1 or  $L_1$  from candidate patterns of length 1 or  $C_1$  (all possible unique symbols in the input sequence). It then generates higher level candidate patterns ( $C_i$ ) from immediate previous level frequent patterns or  $L_{i-1}$ . Thus  $C_2$  is generated by self joining  $L_1$  with itself (i.e  $C_2 = L_1 * L_1$ ). Subsequent frequent patterns ( $L_i$  or  $L_2$ ) are generated by subjecting the corresponding level candidate patterns to the support factor. Since a complete scan of the database is required for a candidate set to be identified as frequent or

infrequent, the number of repeated scans of the original database during the entire FPM process is huge and is a serious limitation.

With respect to conventional frequent set(pattern) mining, the repeated scans limitation of Apriori has been overcome by several algorithms [8]. The Frequent Pattern (FP) tree based algorithm [5] requires only two overall scans of the original input and adopts a tree or pattern growth approach. It is not suited to FTP mining since it loses track of the ordering of symbols within a pattern. Patterns AB and BA are identical in FPM but in FTP mining they are treated as different patterns. The Dynamic Item-Set Counting approach [7] reduces the number of scans by constructing frequent sets in a simultaneous fashion. Frequent set mining algorithms discussed in [8] either lack the temporal or ordering property or require several repeated scans of the input sequence when used for FTP mining and hence there is an impending research requirement for efficient algorithms for the same.

FTP mining, can also be treated as Sequence Pattern Mining(SPM). GSP [19], an apriori principle based SPM algorithm does not meet our requirement of avoiding the huge repeated scans setback. A few of the other SPM algorithms avoiding the repeated scans setback of Apriori are FreeSpan [4], PrefixSpan [3] and SPADE [6]. FreeSpan and PrefixSpan adopt a pattern growth approach similar to FP growth but incorporate ordering aspect in the mining process. The database projection logic used in these algorithms do require repeated scans, but it is significantly lesser than Apriori. Also the scans are not over the entire database but over a trimmed version. These algorithms are not suited to FTP mining, since they require the input to be in the form of transactional records. Transactionalizing the input sequence to a record format results in loss of temporal continuity across records. Hence there is an impending research requirement of efficient algorithms for mining frequent temporal patterns. The next section presents the proposed FTP mining algorithms, avoiding the repeated input scans and meeting the temporal continuity restrictions.

## 4 Proposed Algorithms for Frequent Temporal Pattern Mining

This section presents the three algorithms proposed for FTP mining that overcome the repeated input scans limitation of Apriori and hence contribute to efficient video association mining approaches [20, 21, 22]. The algorithms accept as input the shot cluster sequence output of the transformation phase and generate the various possible frequent patterns subject to the temporal support and distance threshold factors. The permutations based FTP mining algorithm (p-FTP) is described in Figure 1. Algorithm t-FTP that adopts a pattern growth approach is explained in Figure 2 while Figure 3 explains the m-ary tree based FTP mining algorithm. Illustrations are not included due to space constraints.



1. Scan the original input sequence once to identify frequent and infrequent single length item sets (patterns) by subjecting the candidate single length patterns to the minimum support threshold factor. Frequent single length patterns constitute the set  $L_1$ . Assume  $|L_1|$  is  $k$ .
2. Generate all possible permutations of patterns using elements of  $L_1$ .
3. Order the various patterns generated in Step 2 into  $k$  distinct groups. Grouping is done on the basis of starting label of the respective patterns. Thus if  $L_1$  is composed of 3 symbols namely A, B and C, then group A would comprise patterns A, AB, AC, ABC and ACB. On similar lines other patterns commencing with B and C can as well be grouped respectively. Every pattern has an associated count parameter (initialized to 0) to reflect the count or occurrence of the pattern in the input sequence.
4. For every element or symbol of the input sequence:
  - (a) Establish the group to which the element belongs to.
  - (b) Increment the counter of single symbol pattern of the respective group suitably.
  - (c) For other patterns of the group established in 4.a, count is incremented only when the entire pattern is encountered in the input sequence. In other cases, the starting labels are disabled (denoting that the respective symbol is encountered in the input arrived so far).
  - (d) If starting labels have been disabled, patterns of the respective group being processed are reordered with
    - i. other groups identified in Step 2 for the first input element.
    - ii. other groups identified in earlier iteration.
 Regrouping is required since after disabling, starting labels of patterns differ.
  - (e) Non single length patterns have their counts incremented only when they match as an entire pattern (all symbols have been disabled). Once its count is incremented, it is reintroduced into the respective group, with disabled symbols enabled again.
5. The minimum support threshold criterion is applied to the various patterns by the end of Step 4, resulting in the net FTP set.

**Fig. 1.** Proposed p-FTP Mining Algorithm

## 5 Complexity Analysis of the Algorithms

### 5.1 Apriori Based FTP Mining - Time Complexity

Two key phases of Apriori based FTP mining are candidate patterns generation and support computation of the generated candidate patterns. Let us assume that the length of the original input sequence is  $N$ . The candidate pattern generation process is of the order of  $(C_{m_j}^2 \cdot O(j))$ , where  $C_{m_j}$  denotes candidate set composed of  $m_j$  subsequences. Generation of all frequent sequences of length  $k$  requires  $k$  loops over the original input sequence. Thus the searching phase is of the order of  $kN$ . Support computation process for  $C_j$  that contains  $m_j$

1. Scan the original input sequence to identify frequent 1 items that constitutes the set  $L_1$ .  $L_1$  is generated by subjecting the candidate patterns of length 1 to the minimum support threshold factor.
2. For every member of  $L_1$ , now construct a pattern tree as follows:
  - (a) Make the element of  $L_1$  the root of the tree.
  - (b) Start scanning the original input sequence from the point where the considered element appears.
  - (c) Add paths from the root of the tree to the item encountered in the sequence in a cumulative fashion updating both the item count as well as the path count. This is done till the considered element appears again in the input (a subsequence). If an element that has already been added as a node to the tree appears again then maintain suitable reverse links as well.
  - (d) Repeat the earlier step for other sequences that commence with the considered element, for which either traverse the existing paths updating the counters suitably or add new paths from the root of the tree depending on the items encountered in the sequence.
3. From each of the pattern tree's constructed, temporal patterns and hence frequent temporal patterns are generated as follows:
  - (a) Traverse the tree from the root retrieving patterns whose temporal count is the path count.
  - (b) Once a branch is traversed, suitably decrement the path count as well as the item count to reflect that the pattern has been counted once.
  - (c) To generate patterns that do not appear as a branch in the tree, consider pending nodes that appear to the right of some specific node in the tree as either destination or intermediate items in the pattern.
  - (d) Integrate such patterns that match with the tree branch pattern to generate the net temporal pattern set .
4. From each of the pattern tree's temporal pattern set, retain only those patterns that satisfy the temporal support threshold specified.

**Fig. 2.** Proposed t-FTP Mining Algorithm

subsequences is of the order of  $m_j \cdot O(j \cdot N)$ . For example, support computation for candidate set  $C_1$  requires  $m_1 \cdot O(N)$  time , where  $m_1$  is the cardinality of  $C_1$ . Thus the overall time complexity of Apriori based FTP mining algorithm is given by  $T_a = \sum_j \left( (C_{m_j}^2 \cdot O(j)) + (m_j \cdot O(j \cdot N)) \right)$ .

## 5.2 p-FTP - Time Complexity

This section discusses the time complexity of the proposed p-FTP algorithm. Let  $N$  denote the length of the original input sequence and  $|L_1|=n$ . Assume  $M$  represents the total number of candidate patterns generated by permuting the  $n$  frequent 1 elements and  $m_j$  denotes the maximum number of candidate

1. Scan the original input sequence once to identify frequent 1 items that constitutes the set  $L_1$ .  $L_1$  is generated by subjecting the candidate patterns of length 1 to the minimum support threshold factor. Let  $|L_1|$  be  $m$ .
2. Construct a m-ary tree of height  $m$  with root as empty node ( $e$  or  $\epsilon$ ).
3. Label the  $m$  descendant paths of all nodes in the tree at a particular level with the different possible symbols in  $L_1$ . Maintain two parameters for every path in the tree namely count and traversed that will be updated during the algorithms execution.
4. For the first input sequence element, the respective path from  $e$  or  $\epsilon$  that bears the same label as the element is traversed, its count parameter is incremented and the visited parameter is enabled.
5. For every other input sequence element:
  - (a) The respective path from  $e$  or  $\epsilon$  with the same label as the considered element is traversed, suitably updating the count and visited parameters.
  - (b) Also other paths in the tree whose visited parameters have already been enabled are traversed. Count parameter is suitably incremented and visited parameter is enabled if they are not already enabled for paths whose labels match with the input element encountered. Since repetitive patterns are as well maintained due to the completeness of the algorithm, they are handled as a special case. Counts at leaf path levels are incremented only when corresponding count at root path level has been incremented as a multiple of the tree depth.
6. To generate frequent patterns, the m-ary tree at the end of step 5 is traversed retaining those paths that satisfy the minimum support threshold.

**Fig. 3.** Proposed m-FTP Mining Algorithm

sets of length  $j$  processed within any group. Permutations are generated in a time efficient manner using exchanges method of linear time complexity [23]. Grouping of candidate sets done in Step 3 of the algorithm is of the order of  $\sum_j m_j \cdot O(j)$  and filtering of frequent patterns done in Step 5 requires  $O(M)$  time. Step 4 is of the order of  $\sum_i \sum_j m_j \cdot O(j)$ . Thus the overall time complexity of p-FTP is given by  $T_p = O(n^2) + \sum_i \sum_j m_j \cdot O(j) + O(M)$ .

### 5.3 m-FTP - Time Complexity

Time complexity of m-FTP involves the time required for complete m-ary tree construction, update of the m-ary tree during the processing of input sequence elements and frequent temporal patterns generation. Assume the original input sequence is of length  $N$  as before. The height of a complete m-ary tree is  $O(\log_m n)$ , where  $n$  is the total number of leaves in the tree. Number of leaves at a particular depth  $d$  is  $m^d$ . Based on these factors, the complete m-ary tree construction phase is of the order of  $n \log_m n$ , where  $n$  is the number of leaves.

The tree is updated by traversing branch(es) that match with the input sequence element arrived or those that have already been traversed. Not all branches of the complete  $m$ -ary tree are traversed during the processing of an input sequence element. Assuming that  $b$  is the number of branches traversed during the processing of a specific input sequence element, the update process is of the order of  $bN$ . Retrieval of patterns or tree traversal subject to the support factor is of the order of  $n \log_m n$ . Thus the overall complexity of the proposed algorithm is given by  $T_m = O(n \cdot \log_m n) + O(bN)$ .

t-FTP algorithm's complexity is dependent on the actual makeup of the sequences(subsequences or sub transactions) and its improvements in execution time is established using experimental results in the following section.

#### 5.4 Space Complexity Analysis

p-FTP constructs all possible permutations of patterns using elements in  $L_1$ . For a sample input consisting of 3 unique elements, p-FTP processes with 15 candidate patterns. In general for  $n$  unique elements, the total space required works out as  $n!$ . Number of candidate patterns in Apriori approximates to combinations of the number of frequent patterns in the immediate prior level(1) and hence is  $!$  times lesser than the space consumed by p-FTP.

In m-FTP, any particular level  $d$  in the tree can have a maximum of  $m^d$  states or patterns and hence the space consumed is exponential in nature. Though the space complexity of p and m-FTP are more in comparison to Apriori (a tradeoff as a result of the reduced input scans), Apriori would also suffer from the same problem when the number of frequent patterns at any particular level is large. As a result of the recent advances in storage technology and extremely huge memory capacities to the tune of giga and terra bytes available these days, performance improvements in time are always preferred and crucial.

Thus the proposed approaches, namely p and m-FTP make significant contributions to the temporal frequent set mining community despite their space limitations. t-FTP adopts an on the fly approach to the entire frequent temporal pattern mining process. Pattern trees are grown based on the input sequence element encountered and there is no exhaustive generation of candidate patterns as opposed to the other two proposed approaches. This results in a balanced algorithm that offers considerable improvement in time in relation to Apriori and is also a space efficient version as explained in the next section.

## 6 Experimental Results

This section deals with performance analysis of the proposed algorithm in comparison to Apriori and the proposed approaches. All experiments have been conducted on a 256MB RAM machine supporting LINUX 9. Algorithms are analysed for the effect of sequence length and support factor on execution time. Figure 4 establishes the performance improvement of the proposed algorithms as a result of the reduced number of input scans in comparison to Apriori. A

similar qualitative growth pattern is observed with other varied input sequences. We have observed 73%, 62% and 53% improvements in execution time of m, p and t FTP mining algorithms respectively in relation to Apriori based one.

m-FTP achieves the best performance due to the significantly reduced number of scans (two) in comparison to Apriori and t-FTP. Between m and p-FTP, the time used up in computing all possible permutations and frequent enabling/disabling logic in p-FTP results in the performance variation. p-FTP outperforms t-FTP and Apriori primarily as a result of the reduced number of input scans. t-FTP is second to m and p-FTP in terms of performance improvement (due to more repeated (but significantly less compared to Apriori) scans than p and m-FTP). However t-FTP achieves a balance in the space time complexity tradeoff. Since it is a pattern growth approach, it is only the patterns or sequences encountered in the input that are maintained by the algorithm, as opposed to the exhaustive approaches adopted by m and p-FTP. Thus the space requirements of t-FTP is significantly lesser compared to Apriori, m and p-FTP.

Despite being low on space consumption, t-FTP achieves better performance in comparison to Apriori and moderately lowered performance in comparison to m and p-FTP, as a result of the more number of repeated scans. Thus situations

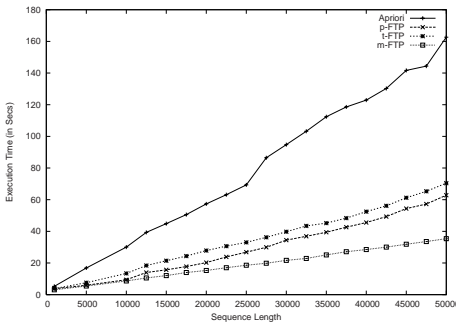


Fig. 4. Effect of Sequence Length

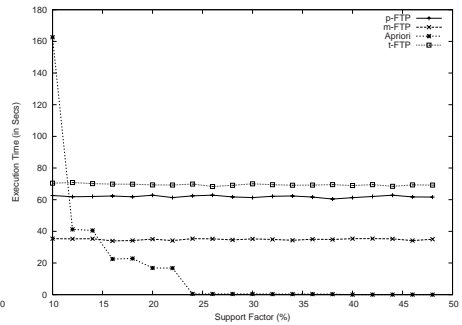


Fig. 5. Effect of Support

where the input is composed of lengthy sequences involving relatively lesser number of unique elements and more frequent patterns, algorithms m and p-FTP are preferred in comparison to Apriori and t-FTP. On the other hand, for inputs involving lengthy sequences and more unique elements, t-FTP is ideal compared to Apriori, m and p-FTP. This is due to the balanced space and time improvements offered by t-FTP. In the case of inputs involving lengthy sequences, more unique elements and less number of frequent patterns, Apriori dominates the rest as a result of its support biased logic, wherein it prunes out infrequent patterns at a much earlier stage of the algorithm in relation to the proposed approaches. The support based logic favours less frequent patterns scenarios, but also contributes to the repeated scans limitation of Apriori with inputs composed of more frequent patterns.

The proposed algorithms are independent of the support factor as opposed to the level wise support dependent logic of Apriori. Execution time of Apriori is inversely proportional to support threshold and hence execution times at lowered support values are high compared to higher support values. This is a result of Apriori's level wise principle, wherein, as a result of the reduced support values, more patterns become frequent at a particular level. Candidate and subsequent frequent pattern generation becomes time consuming due to the increased frequent patterns in the preceding level. The effect of support threshold on execution times of the algorithms is shown in Figure 5. As can be observed, all the three proposed approaches exhibit only slight variation in execution times due to run time resource allocation constraints. On the other hand, Apriori shows considerable variation in execution times due to the changing support factors. Varied input sequences establish a similar qualitative behavior. The proposed approaches require the support parameter only during the initial scan to decide on frequent and infrequent single length patterns and finally to generate all possible frequent temporal patterns. Apriori on the contrary, requires the support factor at each level to generate frequent patterns from candidate patterns.

## 7 Conclusion

Frequent Temporal Pattern mining is an essential phase of video association mining. We have studied the algorithms for FTP mining comparing the existing Apriori based one with our proposed approaches. Association Mining research in the domain of conventional databases led to the evolution of several efficient and application specific frequent pattern mining algorithms. FTP mining is a relatively new and emerging research area with the potential for further efficient and application specific approaches. The proposed algorithms overcome the repeated scans limitation of Apriori and hence contribute to an efficient video association mining strategy. Case specific applications of generated frequent patterns such as classification, summarization and dedicated algorithms for the same is a candidate for further research.

## References

1. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kauffman. (2001)
2. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Very Large DataBases (VLDB). (1994) 487–499
3. Han, J., Pei, J. et.al: Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. IEEE Transactions of Knowledge and Data Engineering. **16**(11) (2004)
4. Han, J., Pei, J. et.al: Freespan: Frequent Pattern Projected Sequential Pattern Mining. In: ACM SIGKDD International Conference on Knowledge Discovery in Databases. (2000) 355–359
5. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns Without Candidate Generation. In: ACM SIGMOD Conference. (2000) 1–12

6. Zaki, M.: SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, **40** (2001) 31–60
7. Brin, S., Ullmann, J., Motwani, R., Tsur, S.: Dynamic Itemset Counting. In: *ACM SIGMOD Conference*. (1997) 255–264
8. Goethals, B.: Survey on Frequent Pattern Mining. Helsinki Technical Report. **43**(2003)
9. Zhu, X., Wu, X.: Mining Video Associations for Efficient Database Management. In: *18<sup>th</sup> International Joint Conference on Artificial Intelligence*. (2003) 1422–1432
10. Zaiane, O.R., Han, J., Li, Z., Chiang, J.: Multimedia Miner—A System Prototype for Multimedia Data Mining. In: *ACM SIGMOD Conference*. (2002) 581–583
11. Zaiane, O.R., Antonie, M.L., Coman, A.: Mammography Classification by an Association Rule based Classifier. In: *International Workshop on MDM with ACM SIGKDD*. (2002) 62–69
12. Zhu, X., Wu, X.: Sequential Association Mining for Video Summarization. In: *ICME, Baltimore*. (2003) 333–336
13. Tesic, J., Newsam, S., Manjunath, B.S.: Mining Image Datasets using Perceptual Association Rules. In: *Mining Scientific and Engineering Datasets*. (2003) 71–77
14. Zaiane, O.R., Han, J., Zhu, H.: Mining Recurrent items in Multimedia with Progressive Resolution Refinement. In: *International Conference on Data Engineering*. (2000) 461–470
15. Wijesekara, D., Barbara, D.: Mining Cinematic Knowledge—Work in Progress. In: *International Workshop on MDM/KDD*. (2000) 98–103
16. Zaiane, O.R., Han, J., Li, Z., Hou, J.: Mining Multimedia Data. In: *CASCON*. (1998) 83–96
17. Zaiane, O.R.: Resource and Knowledge Discovery from the Internet Multimedia Repositories. PhD thesis, Simon Fraser University. (1999)
18. Zhu, X., Wu, X. et.al: Video Data Mining : Semantic Indexing and Event Detection from the Association Perspective. *IEEE Transactions of Knowledge and Data Engineering*, **17**(5)(2005)
19. Srikant, R., Agrawal, R.: Mining Sequential Patterns – Generalizations and Performance Improvements. In: *5<sup>th</sup> International Conference on Extending Database Technology (EDBT)*. (1996) 3–17
20. SivaSelvan, B., Krishnamurthi, I.: An Efficient Video Association Mining Algorithm. *Journal of Computer Society of India*, **35**(3) (2005) 56–67
21. SivaSelvan, B., Gopalan, N.P.: An Efficient Frequent Temporal Pattern (EFTP) Mining Algorithm. *Information Technology Journal*, **5**(6) (2006) 1043–1047
22. Gopalan, N.P., SivaSelvan, B.: An m-ary tree based Frequent Temporal Pattern (FTP) Mining Algorithm. In: *6<sup>th</sup> IEEE INDICON International Conference*. (2006)
23. Sedgewick, R.: Permutation Generation Methods. *ACM Computing Surveys*, **9**(2) (1977) 137–164

# Distributed Data Mining in a Ubiquitous Healthcare Framework

M. Viswanathan

Carnegie Mellon University - Australia  
H. John Heinz III School of Public Policy and Management,  
Adelaide, SA, Australia  
murli.viswanathan@gmail.com

**Abstract.** Ubiquitous Healthcare (u-healthcare) which focuses on automated applications that can provide healthcare to citizens anywhere/anytime using wired and wireless mobile technologies is becoming increasingly important. Ubiquitous healthcare data provides a mine of hidden knowledge which can be exploited in preventive care and “wellness” recommendations. Data mining is therefore a significant aspect of such systems. Distributed Data mining (DDM) techniques for knowledge discovery from databases help in the thorough analysis of data collected from healthcare facilities enabling efficient decision-making and strategic planning. This paper presents and discusses the development of a prototype ubiquitous healthcare system. The prospects for integrating data mining into this framework are studied using a distributed data mining system. The DDM system employs a mixture modelling mechanism for data partitioning. Initial results with some standard medical databases offer a plausible outlook for future integration.

## 1 Introduction

The advances in and rapid deployment of wireless technology through devices and mobile telephony have resulted in the development of novel ubiquitous healthcare systems that simplify the monitoring and treatment of patients [1]. The provision of healthcare services at any time and any place for individual consumers has become a necessity due to the increase in aging population in several countries and the healthcare field facing strong pressures to reduce costs while increasing quality of services delivered. Such u-Healthcare systems also provide enhanced services including real-time patient data collection and monitoring for emergency care as well as preventive health recommendations.

The vast amount of data collected from the distributed users of u-Health services results in a growing need for analyzing them across geographical lines using distributed and parallel systems. The success of these u-Healthcare systems will depend on smart utilization of healthcare information systems for decision support and knowledge management. In this paper, we focus on the use of a preliminary distributed data mining (DDM) system [12] within a u-Healthcare framework. A data partitioning technique



based on mixture modeling [13] is employed in the distribution of data for local data mining. Subsequently we introduce and present empirical analysis on some simple databases. The indispensability of DDM systems to u-Healthcare is substantiated.

## 2 Ubiquitous Healthcare Initiatives and Challenges

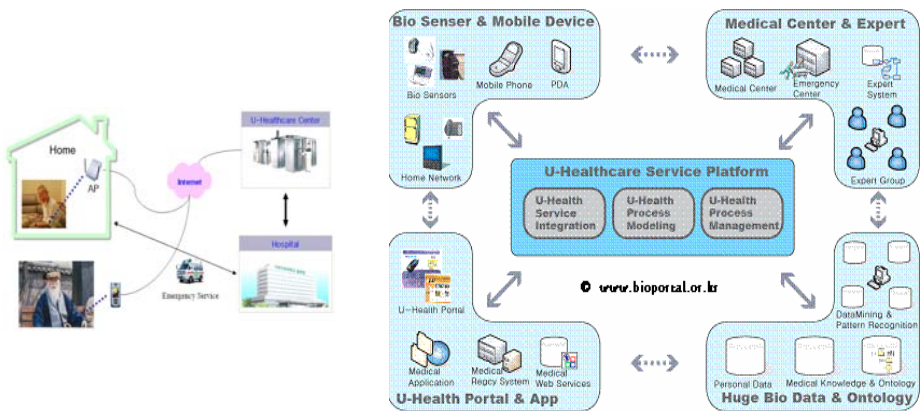
A growing number of ubiquitous healthcare projects are being pursued by large enterprises owning healthcare related companies and government bodies. MobiHealth project [2] is a mobile healthcare project supported by the EC with countries such as Netherlands, Germany, Spain and Sweden participating in it, and companies such as Philips and HP are providing technical support. EliteCare[3], is an elderly care system developed in the USA that monitors patients using various sensors and provides emergency and health information services. Tele-monitoring service[4] is being developed by the Philips Medical system, where centers analyze data that is collected from homes and transmitted by biomedical signal collection devices, and provide health management and related information. CodeBlue[5] is a sensor network based healthcare system being developed to treat and deal with emergencies, rehabilitation of stroke patients, and in general, to use health signal data in addition to hospital records in real time treatment decisions. The UbiMon[6] project which stands for Ubiquitous Monitoring Environment for Wearable and Implantable Sensors is studying mobile monitoring using sensors and real-time biomedical data collection for long time trend analyses. The Smart Medical Home[7] aims to develop a fully integrated personal health system with ubiquitous technology based on infrared and bio sensors, computers, video cameras and other devices. Sensor data is collected and transmitted to a center for further analysis and preventive care.

There are several ubiquitous challenges in the development of such healthcare frameworks and systems. These include:

- issues of security and privacy related to information transfer through unsecured infrastructure, potentially lost or stolen devices, legal enforcement and other scenarios;
- determining current context and user activity in real-time and locating context dependent information such as automatic discovery of services based on user health needs;
- development of low-power sensors to monitor user context and health condition;
- information management through development of techniques to collect, filter, analyze and store the potentially vast quantities of data from widespread patient monitoring and applying privacy preserving data mining at several levels;
- simple patient interaction systems to provide guidance, feedback and access to medical advice in acute situations;
- Adaptable network infrastructures to support large-scale monitoring, as well as real-time response from medical personnel or intelligent agents.;
- integration of specialized local u-Health architectures for unified data access and connection to National grids;

### 3 Ubiquitous Healthcare System Framework

Figure 2 shows an overview of the ubiquitous healthcare service framework as suggested in this paper. A system user in this paper refers to a patient who has a contract with a provider to use the ubiquitous healthcare services and regularly receives medical treatment at a hospital. The user wears a sensory device, provided by the hospital, on his wrist. The sensor regularly transmits collected data to a healthcare center through networking or mobile devices, and the transmitted data is stored at the u-healthcare center. In the center, monitoring staff are stationed to answer the user's queries, monitor his biomedical signals, and call an emergency service or visit the patient to check his status when an abnormal pattern is detected. The hospital monitors the collected data and judges the patient's status using the collected biomedical signals in his periodic checkup. It is important to note the framework is currently developed for a subscription-oriented service.



**Fig. 1. System Framework**

### 3.1 Biomedical Signal Collection and Transmission

The wrist sensor, attached to a user's wrist throughout the day, collects data such as the user's blood pressure, pulse, and orientation and transmits the collected data to the user's mobile phone or access point (AP) at home using a wireless ZigBee device. ZigBee is established by the ZigBee Alliance and adds network, security and application software to the IEEE 802.15.4 standard. Biomedical signals can be collected while moving in and out of the user's residence. The data collected inside of the house is sent to the AP in the house using Zigbee module. The AP stores the collected data and sends it regularly to the data storage at the healthcare center. When the user is outside of the house, the sensor sends the collected data to the user's mobile phone and then using CDMA module of the mobile phone, transmits the data to the center.

A *light-weight data mining* component is being developed for the mobiles and APs which briefly analyzes the data collected. This component has the responsibility of judging if an emergency occurs by analyzing the biomedical signals collected by the

sensor. It also includes a function to call an emergency service using a motion detector attached to the sensor if it detects user collapse.

### 3.2 Healthcare Center

The healthcare center has two primary roles. First, it provides storage and management for the biomedical data collected from the users, and second, it monitors the users' health status and takes appropriate emergency or preventive action when required. A database server in the healthcare center stores and manages data including the medical, personal, family and other information for all registered users as well as biomedical signals collected from them. This data is used for real-time monitoring of users in case of emergencies and is also useful in periodic checkups.

The healthcare center also includes personnel who are stationed to keep monitoring users' health status and provide health information as well. Some of their responsibilities include regular phone checks, personal visits to users and emergency assistance if any abnormal signals are detected from a user.

### 3.3 CDSS (Clinical Decision Support System)

We plan to provide the following decision-support/strategic-planning services including, analysing trends in hospital admission, analyzing treatment pattern, analyzing outcomes of treatment, analysing cost-effectiveness of health care, planning out-of-hospital (ambulatory) care, forecasting 'new disease' and strategizing appropriate preventive measures, forecasting complications of treatment, forecasting the spread of infectious diseases. The CDSS supports long-term and short-term decision making processes by using models from distributed data mining, developing alternative plans and performing comparison analysis. In the short-term it assists in optimal planning to solve various decision making problems confronted in emergencies by utilizing the biomedical signals. The goal of this system is to provide an information system environment where a decision maker can solve problems easily, accurately and promptly such that users are benefited. The CDSS needs to be integrated with a distributed data mining system that can provide global models.

### 3.4 Emergency Response

Emergencies in a U-health framework require robust and quick recognition followed by an efficient emergency response. In this framework we employ a three pronged emergency recognition drive. Firstly, personnel monitoring the streaming biomedical data may detect abnormal signs and check user through phones or visits. Secondly, abnormal signs are also detected while mining the biomedical data collected over a period by the CDSS. Lastly, motion detectors mounted on sensors detect occurrence of falls and erratic movement.

The emergency management system uses a variety of hardware and software components that aim to improve emergency counteractions at the appropriate time and lower preventable deaths. This includes portable personal terminals comprising of RFID tags, portable RFID readers, an ambulance information system, a hospital information system and a healthcare information system. The efficiency of the treatment in emergency rooms is increased by using RFID tags and readers. Since the system is well integrated

it also transfers patient information in real-time to hospitals, and therefore medical teams who will provide treatment during emergencies can be well-prepared.

### 3.5 Remote Monitoring System

With increasing urbanization, shrinking of living space and shifting concepts of the family, elderly people often tend to live alone without any assistance at home. In such cases prompt responses are most important when a medical emergency occurs. The remote monitoring system is used to detect falls and erratic movement occurring at homes remotely using cameras or by checking current situations when an abnormal sign is detected. There may be signals that cannot be detected even with motion detectors mounted on sensors, or false alarms may occur. In these cases, the situations can be checked using in-house video cameras. The remote monitoring system is not only a management system for patient monitoring but aims for general health improvement of consumers through prevention of diseases, early detection, and prognosis management. Thus a customized personal healthcare service is established, maintained and controlled continuously [9, 10].

## 4 Distributed Data Mining

Distributed Data Mining (DDM) deals with mining of geographically distributed data. DDM research is continually developing improved tools and methods to deal with distributed data. Databases in u-healthcare domains are naturally distributed geographically and data from all sites must be used to optimise data mining models. There are many different DDM systems, algorithms, techniques and paradigms. In our study we address the case where a hospital or health data center has a large local database which needs to be analysed on a local cluster.

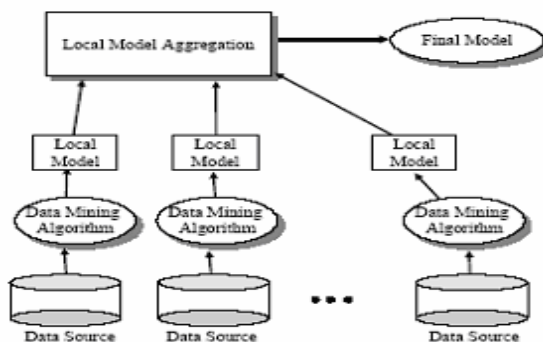


Fig. 2. A Typical DDM Scenario

### 4.1 Cluster-Based DDM

There are a large number of aspects that need to be studied when designing cluster-based DDM systems. In general the following steps are significant:

1. Data Partitioning— a large dataset must be partitioned into smaller subsets of data with the same domain according to some function  $F$ .
2. Data Distribution – the individual partitions must be distributed for processing over the cluster of PCs typically over the LAN. An appropriate distribution scheme may be devised depending on number of clusters, size of clusters, content of clusters, size of LAN and LAN communication speeds.
3. Data Modelling – after distribution local data mining models have to be constructed on the nodes in the cluster. Many different knowledge discovery tools and algorithms are widely available. The choice of classification algorithm depends on the nature of the data (numerical, nominal or a combination of both), the size of the data and the destined nature and purpose of the models.
4. Model Aggregation - local models must be combined and aggregated in an optimal manner to produce a global model that supports accurate decision support. Model aggregation attempts to achieve one of the fundamental aims of DDM that is to develop a final global model from the distributed data to match the efficiency and effectiveness of a data model developed from undistributed data.

## 4.2 Mixture Modelling in Data Partitioning

Local healthcare databases are often too large to process on one machine due to cpu, memory and space restrictions. Therefore we require tools and techniques to partition the database into manageable subsets. SNOB is a system developed for cluster analysis and automatic classification using mixture modelling by Minimum Message Length (MML) [13]. SNOB aims to discover the natural classes in the data by categorising data sets based on their underlying numerical distributions. It does this using the assumption that if it can correctly categorise the data, then the data can be described most efficiently (i.e. using the minimum message length). SNOB uses MML induction, a scale-invariant Bayesian technique based on information theory. A related goal in our project is to investigate whether using SNOB as our method of choice for clustering is appropriate and efficient in the framework of DDM.

A database is usually composed of many objects or instances. Each instance has a number of different attributes with each attribute having a particular value. We can think of this as a population of instances in a space with each attribute being a different dimension or variable. SNOB assumes to know the nature, number and range of the attributes. The attributes are also assumed to be uncorrelated and independent of each other. SNOB attempts to divide the population into groups or classes such that each class is tight and simple while ensuring the classes' attribute distributions significantly differ from one another. In figure 4 we can see the partitioning part of our DDM system with SNOB being the core of the partitioning operation. SNOB is used to cluster the data and generate a report file, which is then scanned and the complete database gets divided into the appropriate partitions.

In brief SNOB generates a hypothesis (the first part of the general message) that is translated into the data's class structure. The data is coded optimally by shortest message length where the hypothesis holds true (this is the second part of the general message and is talked about above). Note that SNOB doesn't construct the message but rather only calculates the length the message would have if it were constructed. The message length is calculated in 'nits', a nit is  $[\log e / \log 2]$  and is a unit message length.

## 5 Healthcare Decision Support with Distributed Data Mining

A small number of published studies address the value of data mining within the healthcare industry. ANNs have been used to predict transfusion needs [21], identify myocardial infarctions [1], estimate drug and plasma concentrations levels of pharmaceutical drugs [20], and predict the risk of coronary artery disease [12]. All studies assert that a key strength of ANNs compared to traditional statistical models is their ability to deal with nonlinearities in data sets while not worrying about the underlying distribution of data [5]. Other popular data mining techniques applied to healthcare are Bayesian models, association rules, case-based reasoning, genetic algorithms, and fuzzy systems (see[18] for applications).

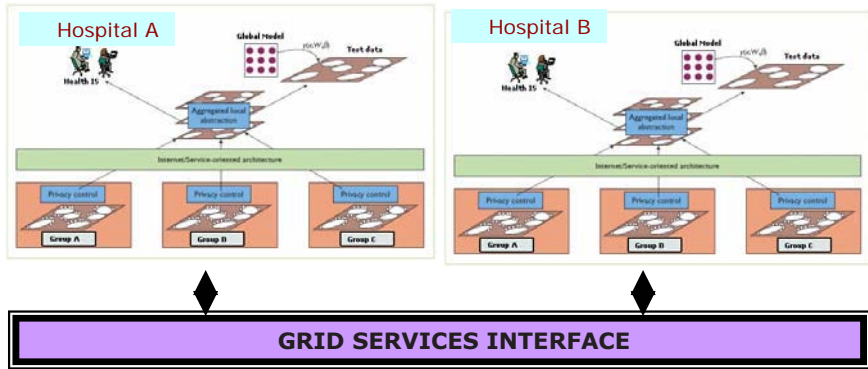
**Table 1.** Levels of u-Healthcare data mining

Low Level	sensor data mining for emergency detection
	sensor data fusion for decision support
Hospital-Based	periodic trend analysis & wellness recommendations
	analysis of patient groups for drug effects, CRM and indirect factors
	recovery analysis, drug interaction and other studies like allergy cause and detection
Global Level	clustering and classification analysis for epidemics and other implicit patterns
	GRID based mining of biomedical data like gene expressions and MRI images, on a National and International scale.

In table 1 we define the various levels of data mining applicable is such healthcare frameworks. In this paper we consider the scenario [12] where a hospital needs to analyze a very large database with real-time patient sensor data. Due to processing and memory limits the database is partitioned and sent to individual machines to be processed. In this section we discuss how distributed data mining plays an important role within the CDSS component of the ubiquitous healthcare system.

### 5.1 CDSS and DDM

In a ubiquitous healthcare framework DDM systems are required due to the large number of streams of data that have a very high data rate and are typically distributed. These need to be analyzed/mined in real-time to extract relevant information. Often such data come from wirelessly connected sources which have neither the computational resources to analyze them completely, nor enough bandwidth to transfer all the data to a central site for analysis. There is also another scenario where the data collected and stored at a center needs to be analyzed as a whole for creating the dynamic profiles. The preliminary empirical analysis with the prototype distributed data



**Fig. 3.** From Local to Global Mining

mining system discussed in this paper is suited towards this latter situation. The integration of the CDSS component of the ubiquitous healthcare framework with such a DDM is important.

Data mining techniques used in the decision making system divide patients into groups. As a collection of patients have their own characteristics, they should be divided properly, and group properties are found through applying cluster analysis modeling techniques and searching created groups in the group analysis step. Secondly, causal models are developed for health patterns using mining techniques. Finally, a dynamic profile of the patient can be created using past history and domain knowledge in conjunction with sensory data. Each patient's risk rate is calculated by a system reflecting mining results, and administrators can see patients' risk rankings from the risk rates and give priority to patients with higher rates.

## 5.2 Distributed Data Mining Architecture

This section describes a prototype system for DDM. For a detailed exposition of this system see [16]. The DDM system is build from various components as seen in figure 3. The DDM system takes source data and using SNOB [13], a mixture modeling tool, partitions it to clusters. The clusters get distributed over the LAN using MPI [14]. Data models are developed for each cluster dataset using the classification algorithm C4.5 [15].

Finally the system uses a voting scheme to aggregate all the data models. The final global classification data model comprises of the top three rules for each class (where available). Note that MPI is used in conjunction with the known maximum number of hosts to classify the clusters in parallel using the C4.5 classification algorithm. If the number of clusters exceeds the available number of hosts then some hosts will classify multiple clusters (using MPI). Also the aggregation model scans all Rule files from all clusters and picks the best rules out of the union of all cluster rule sets.

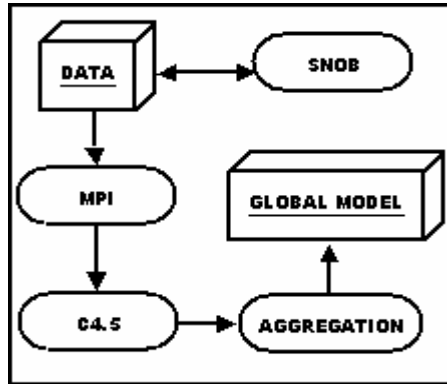


Fig. 4. DDM System Components

During the classification phase we have also classified the original dataset and produced rules modeling this data. To finally ascertain if our DDM system is efficient we compare our global model to this data model from the un-partitioned database. We compare the top three rules for each class from this model with our rules from the global model. If our global model is over 90% accurate in comparison to the data model from the original database we consider this as a useful result.

### 5.3 Preliminary Results

The DDM system was tested on a number of real world datasets in order to test the effectiveness of data mining and the predictive accuracy. Detailed empirical analysis can be studied from [16]. In this section we present the DDM system performance results on the *Pima-Indians-Diabetes* and *Yeast* datasets taken from the UCI KDD Archive [17]. The diagnostic is whether the patient shows signs of diabetes according to World Health Organization criteria.

In order to study the usefulness of the system we compare the top three rules (where available) for each class from the partition-derived classification rules and rules from the original dataset. The aim of this testing is to find out the effect of our clustering process in partitioning, to the efficiency of our classification model and its predictive accuracy. We consider a 10% threshold, average error rates of rules from partitions greater than 10% of that of the corresponding original rules is not useful.

We can observe in figure 5 that the average error rates of rules from partitions and original rules differ within reasonable limits with the average error rate of partition rules staying above the original rules throughout with this gap closing as we approach higher classes. In general the distributed data mining system offers useful performance in the presence of a number of factors influencing the predictive accuracy. However many improvements and further research is needed in order to optimize the DDM system.



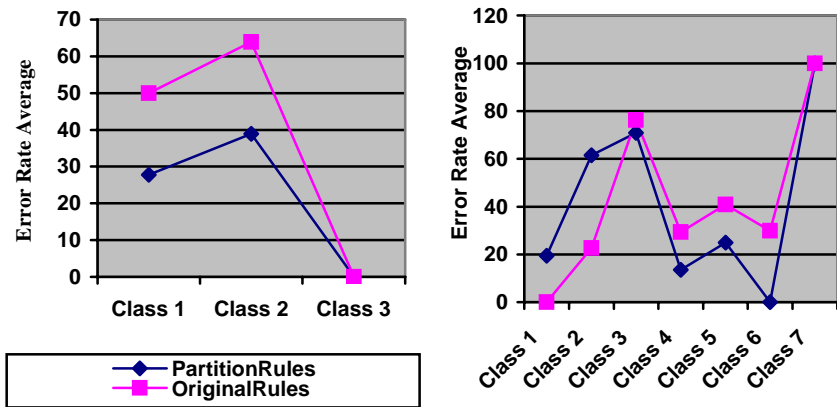


Fig. 5. Predictive Accuracy Comparison

## 6 Conclusions and Future Work

Electronic healthcare (e-healthcare) is now shifting to ubiquitous healthcare which enables real-time monitoring, early diagnosis and treatment for potential risky diseases. In addition to devices such as wearable healthcare sensor systems and smart medical homes, distributed computing for large scale analysis of physiologic signals have also been developed. The growth in wireless technology has also had a significant impact on the development of truly ubiquitous healthcare systems. Provision of quality low-cost healthcare services is becoming an increasingly critical issue due to the elderly population growth in many countries. Our research aims to enable a patient-centric ubiquitous healthcare environment instead of the existing hospital-centric approach.

Despite this advancement in technologies for developing u-Healthcare systems, such systems face an important challenge in maintaining privacy of patient data due to rampant monitoring and access. In the ubiquitous sensor era, it is possible to collect the data from end node and track user locations without their awareness. Further research is needed to counter the security and privacy challenges in u-Healthcare systems. Another issue is that of scalability. It is important to understand that the system described in the paper will offer specialist paid services and is still under development. In many developed countries (e.g., South Korea) network bandwidth is also not a critical issue anymore.

This paper commences by describing a ubiquitous healthcare framework designed to provide consumers with freedom from temporal and spatial restrictions in their access to professional and personalized healthcare services anytime and anywhere – even outside of the hospital. The use of traditional verification-based approaches to analysis is difficult when the data is massive, highly dimensional, distributed, and uncertain. Components of the system framework are discussed in brief. A prototype distributed data mining system based on mixture modelling is introduced with results

from preliminary experiments on data. The plausibility of integrating such a DDM system with the clinical decision support component (CDSS) of the ubiquitous healthcare frameworks is highlighted.

## References

- [1] T. Znati. On the challenges and opportunities of pervasive and ubiquitous computing in health care. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications - PerCom 2005*, page 396, Kauai Island, HI, USA, Mar. 2005.
- [2] MobileHealth: <http://www.mobihealth.org/>.
- [3] EliteCare: <http://www.elitecare.com/index.html>.
- [4] Tele-Monitoring: <http://www.medical.philips.com>.
- [5] CodeBlue: <http://www.eecs.harvard.edu/~mdw/proj/codeblue>.
- [6] Kristof Van Laerhoven et al.: Medical Healthcare Monitoring with Wearable and Implantable Sensors, 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications (2004).
- [7] Smart Medical Home, [http://www.futurehealth.rochester.edu/smart\\_home/Smart\\_home.html](http://www.futurehealth.rochester.edu/smart_home/Smart_home.html)
- [8] J. Hill, M. Horton, R. Kling, L. Krishnamurthy: The Platforms enabling Wireless Sensor Netowrks. *Communications of the ACM*, Vol. 47 (2004) 41-46.
- [9] Jardine, I. and Clough K.: The Impact of Telemedicine and Telecare on Healthcare. *Journal of Telemedicine and Telecare*, Vol. 5, Supplement 1 (1999) 127-128.
- [10] Smithers C. R. and Hill N.: Options for Wireless Technology in Telemedicine and Telecare Applications. *Journal of Telemedicine and Telecare*, Vol. 5, Supplement 1 (1999) 138-139.
- [11] Anup Kumar and Mehmed Kantardzic, "Distributed Data Mining: Framework and Implementations", *IEEE Internet Computing*, vol. 10, no. 4, 2006, pp. 15-17.
- [12] Park B. H., Kargupta H.: "Distributed data mining: Algorithms, systems, and applications". *The Handbook of Data Mining*. Nong Ye (ed) Lawrence Erlbaum, New Jersey 2003.
- [13] Wallace C. S., Dowe D. L.: "MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions". *Statistics and Computing*, 10(1), pp. 73-83, January 2000.
- [14] Message Passing Interface Forum. "MPI: A message-passing interface standard". *International Journal of Supercomputer Applications*, 8(3/4):165-414, 1994.
- [15] Quinlan, J. R.. C4.5: Programs for machine learning. San Mateo, CA: Morgan Kaufmann (1993).
- [16] M. Viswanathan, Y. K. Yang, T. K. Whangbo. Distributed Data Mining on Clusters with Bayesian Mixture Modeling, *Lecture Notes in Computer Science*, Volume 3613, Jul 2005, Pages 1207 – 1216.
- [17] Merz, C. and Murphy, P. (1998). UCI repository of machine learning databases. A.B. Smith, C.D. Jones, and E.F. Roberts, "Article Title", Journal, Publisher, Location, Date, pp. 1-10.
- [18] Margaret R. Kraft, Kevin C. Desouza, Ida Androwich. Data Mining in Healthcare Information Systems: Case Study of a Veterans' Administration Spinal Cord Injury Population, 2000.

# Constructing a User Preference Ontology for Anti-spam Mail Systems

Jongwan Kim<sup>1,\*</sup>, Dejing Dou<sup>2</sup>, Haishan Liu<sup>2</sup>, and Donghwi Kwak<sup>2</sup>

<sup>1</sup> School of Computer and Information Technology, Daegu University  
Gyeonsan, Gyeongbuk. 712-714 South Korea  
jwkim@daegu.ac.kr, jongwan@cs.uoregon.edu

<sup>2</sup> Department of Computer and Information Science, University of Oregon  
Eugene, Oregon 97403, USA  
{dou, ahoyleo, dkwak}@cs.uoregon.edu

**Abstract.** The judgment that whether an email is spam or non-spam may vary from person to person. Different individuals can have totally different responses to the same email based on their preferences. This paper presents an innovative approach that incorporates user preferences to construct an anti-spam mail system, which is different from the conventional content-based approaches. We build a user preference ontology to formally represent the important concepts and rules derived from a data mining process. Then we use an inference engine that utilizes the knowledge to predict the user's action on new incoming emails. We also suggest a new rule optimization procedure inspired from logic synthesis to improve comprehensibility and exclude redundant rules. Experimental results showed that our user preference based architecture achieved good performance and the rules derived from the architecture and the optimization method have better quality in terms of comprehensibility.

**Keywords:** user preference ontology, anti-spam system, data mining.

## 1 Introduction

Spam mail is unsolicited, unwanted email sent indiscriminately, directly or indirectly, by a sender having no current relationship with the recipient [1]. Most software for email clients provides some automatic spam mail filtering mechanism, typically in the form of blacklists or keyword-based filters. This filtering technique was somewhat effective in the beginning, but it gradually declined with accuracy over time because spammers started using personal-sounding subjects to thwart the keyword filters [2]. A variety of machine learning algorithms such as naïve Bayesian classifier (NBC) and support vector machine (SVM) have been used for email categorization task on different metadata. While these anti-spam filters achieve statistically impressive accuracies, they remain prone to false positives (i.e., non-spam or legitimate email tagged as spam, which is called “ham”) and false negatives (spam in your mailbox). More to the point, some email is spam to someone but ham to others in many real situations. For

---

\* Currently a visiting scholar at the University of Oregon, USA.

example, it is possible that the a customer service staff at a credit card company may send many business emails to customers and get feedbacks from some of them. In this case, some customers consider the mail as spam but others find it useful. Users' behaviors to emails vary from one another according to the different personal preferences. Therefore it is meaningful to provide user-oriented anti-spam services based on the preferences. In this work, we have collected user preference information and email responses from a group of college students to train an association and classification mining system. Then we have used the generated rules to define a user preference ontology in a formal language. To show how the ontology can help anti-spam systems, we designed a concept of user preference based anti-spam mail system and did a proof-of-concept implementation. The experimental results indicate that the proposed approach is promising.

The paper is organized as follows. Section 2 introduces some related work. Section 3 describes the data collection and preprocessing. Section 4 covers our ontology construction methodology using association and classification mining. Section 5 presents the proposed architecture and experimental results. Conclusions are given in Section 6.

## 2 Related Research

Recently some works about personal email management system have been proposed. Gray and Haahr suggested personalized, collaborative spam filtering [3]. Personalized collaborative filters deliver the most relevant spam notices to each user from the collection of all spam messages that are reported by members of the network. To implement this personalized collaborative filter, whenever a new spam is classified, a signature should be computed and propagated to those users likely to receive a similar mail in order to consider it spam. This requires that information is compiled and maintained for each user and other users who are in similar groups. The P2P architecture lends itself nicely to such a system. Since this approach depends on other user's signature in the P2P network, it is not stand alone. On the other hand, Ravi et al. proposed personalized email management at network edges [4]. Their artificial neural network based spam filter performs spam and virus filtering at the server's origin and therefore saves network bandwidth. The system has two filters: 1<sup>st</sup> filter recognizes text pattern in email and learns from the pattern, and 2<sup>nd</sup> filter learns images in email. This system is very close to human spam identification. But this methodology has a centralized spam filter which just identifies the spam in one email account and helps it learn and then deletes such spam mails from all other accounts of the same user.

Several anti-spam mail systems considering user preferences are currently operating. Most of them require users' selection about what they accept or not based on the recommendation of the anti-spam system. In these systems, an email that could be a potentially spam but cannot be classified as definitely spam will be stored in a specific area. The intended recipient of the email will receive a web link to the specific area, where any emails held there can be classified. Within the specific area the individual user will be able to classify the email as required. The system will remember the individual user's preference and in the future always transmit or block emails from that particular source to the user's inbox [5]. Another setting included in user preferences specifies the languages in which the ham emails are expected to be

written. Current anti-spam systems considering user preference are summarized; they are mainly based on other users' advice in the same group or other email account information of the same user or user's judgment to accept or not based on the recommendation of the system. However, our goal is to develop a user preference ontology based anti-spam management system which is purely based on user preferences and user responses.

There are two main methods to design any domain ontology: top-down and bottom-up. In the top-down approach, ontology experts determine the concepts and their relationships based on their domain knowledge and intuition. In the bottom-up approach, ontology experts select the important concepts by analyzing data coverage and patterns related to them. Both top-down and bottom up approaches need human involvement, although some automatic tools can reduce manually efforts, e.g., the tools which can acquire ontological knowledge from natural language texts [6]. Text-based learning also can be useful for selecting the keywords for the vocabulary in domain thesauri. In this paper, we focus on finding the relationships between user preferences and their behaviors (i.e., responses to emails). The relationships can be represented as rules (axioms) in the domain ontology.

Data mining is useful in discovering the classification rules but it is hard to tell which rules are useful in terms of comprehensibility and accuracy from probably a great number of rules derived through the mining process. Some rules may be too long or too specific to be useful in ontology construction. There are several works on multi-valued logic in machine learning that can be used for rule minimization and optimization. Files and Perkowski explored the multi-valued logic synthesis (MVLS) method [7]. They described how some concepts of machine learning matched nicely with MVLS and showed how MVLS outperformed both C4.5, the widely used classification algorithm and Espresso, an industry standard logic minimization tool. Also, iterative mining for rules with constrained antecedents was reported [8]. This approach was an iterative algorithm that could exploit mining information gained in previous steps to efficiently answer subsequent queries. In this paper, we suggest a simple rule minimization approach based on logic synthesis inspired from Karnaugh map [9] and data mining to exclude lengthy and redundant rules which are not easy for humans to understand.

### 3 Data Preparation

We collected data for user preferences and responses to the emails from the undergraduate students majoring in computer science and information technology at Daegu University in Korea. The first author has conducted several experiments on content-based email filtering [10]. From his previous experience, we aimed at developing an anti-spam mail system based on personal interests and behaviors from a specific user or user group instead of mere analysis of the contents of the email header and body.

First, we designed a user profile format to represent the user preference and their different types of responses to the emails. Many web mail systems such as Yahoo and Comcast provide registration forms to collect personal information about the users' interests. Similar to these forms, we chose {Age, Gender, RequiredHits, News, Finance, Sports, Adults, TvMovieMusic, Kids, Games, Travel, Shopping,

Jobs, RealEstates} as attributes to be included in our user profile. The Age attribute has 2 options -FS (= freshman and sophomore) and JS (= junior and senior) - as all the participants were college students. The RequiredHits (from now on, RHit) was originally adopted in Spam Assassin [2]. It is defined as how many hits are required before a mail is considered as spam, which indicates the strength of the spam filter that the user expects. However not like Spam Assassin which uses numbers in this option, we use linguistic terms such as Very Weak (VW), Weak (W), Neutral (N), Strong (S), and Very Strong (VS) because we do not consider email contents and linguistic representation is more comfortable to people. However, since weak (W) was never chosen by users participated in this work, we eliminated the W value from the RHit attribute set in the experiment. If the users want a very strong spam filter, for example, they can choose VW for RHit, or S if a relatively weak one is preferred. We include the information of email category (henceafter, ECat) labeled by human expert together with the users preferences and their responses in the data mining process to study how the preferences affects responses.

Second, the feature selection was performed before we mined the data. Feature selection involves searching through all possible combination of features in the candidate feature set to find which subset of features works best for prediction. A few of the mechanisms designed to find the optimum number of features are information gain, mutual information, chi squared test and so on. According to previous works on data mining [11], information gain is a good solution to this problem. We calculated information gains for all 15 attributes in a user profile plus the ECat attribute, and chose several attributes from them. The detail of this procedure is described in Section 5.3.

Third, we conclude that email recipients typically have four kinds of response to incoming emails: Reply, Delete, Store, and Spam. When they have no interest on a mail, they just delete it. If they think a mail is important and valuable to respond, then they reply to the sender. Regardless of replying it or not, they sometimes just hold some mails in mail box because the emails might be useful in the future. Finally when a mail is concerned as spam, most of the users move it to a spam dump box to explicitly mark it. Surely, someone can do both actions such as deletion and moving into a spam box but the other either deletes or moves it. The important thing is that most of users would like to maintain only ham in their mail boxes.

Thus we collected some sample emails, personal information and preferences of participating users, and their responses to the samples. Among all the attributes in the profile, most of them have binary values. For example, {FS, JS} are used for the Age attribute and {male (M), female (F)} are given for the Gender attribute, respectively. And all the attributes concerning the user's interests are given in the true-or-false form. If a user is interested in News, he or she checks true for the attribute. However, multi-valued attributes (VW, N, S, and VS) are used for RHit. And ECat attribute uses 12 values to indicate different email category labels; besides the original 11 labels of category, the Etc label is used if no category is labeled for a specific email. The target variable or the output variable of classification mining, Response, has four categories {Reply, Delete, Store, Spam}.

## 4 Ontology Construction

Ontologies, which can be defined as formal specification of vocabulary of concepts and their relationships, play a key role to define the semantics of information for intelligent systems [12]. To achieve user preference based anti-spam system, it will be helpful to construct a domain ontology which can formally define user behaviors based on their preferences. To do that, we mainly perform three steps. The first step is to use association and classification mining to find relationships (rules) between several users' preferences and their email responses. In the next step, we apply a new rule-pruning procedure eliminating redundant rules and preserving highly comprehensible ones. Translation from optimized rules to axioms in a domain ontology is performed during the final step. The details are described as following.

First, we try to discover association rules between various groups of users and their responses for sample email data. For example, we expected that women usually like shopping and students have strong interests in job recruiting. This intuition was realized after we applied association mining to user preference data set. In the same way, we wanted to find unknown correlations between user profiles and user log files, which include user responses to sample emails. Thus, we chose the typical decision tree algorithm, ID3 [11], to train sample email preference data. Our sample data are composed of mostly binary features and some are nominal features as described in Section 3. So ID3 is suitable to discover representative rules from the data set. After ID3 mining was performed, a decision tree is generated. We can convert the decision tree into rules by describing each path of the tree with a rule. From a root node to internal nodes in each path are considered as antecedent conditions of each rule and the leaf node as a conclusion of each rule. To evaluate which rule is good, we count the accuracy by calculating the proportion of testing instances which match the rules.

Second, we apply a new rule minimization procedure in order to exclude redundant rules and select highly comprehensible ones. Thus we suggest a rule pruning approach inspired from logic synthesis. Karnaugh map (K-map) is well known as a simple and easy method to understand Boolean logic simplification [9]. It is possible to find two or more simplified logic expressions in a K-map. For example, a function  $F(A, B, C) = \Sigma(1, 3, 4, 5, 6, 7)$  is composed of three input variables A, B, and C. This function F has 6 min terms, {001, 011, 100, 101, 110, 111}. Two kinds of logic minimizations are possible as shown in Figure 1(a) and (b).

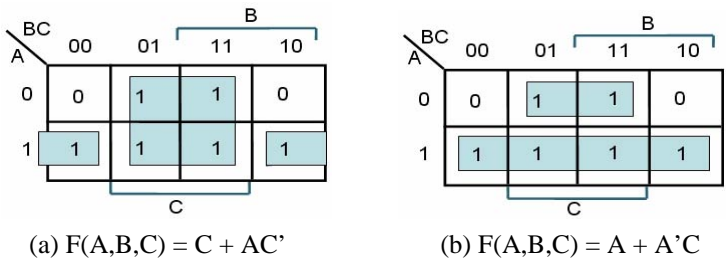


Fig. 1. K-map method examples for 3 variables

As shown in Figure 1, even though the function  $F$  is fixed, it is possible to get two different logic representations. Both expressions are equivalent with respect to logic. Since we got the idea about rule minimization from this K-map example, our rule minimization approach is called logic synthesis based rule pruning (LSRP). There are several variables in a rule set derived from data mining. Most variables are Boolean or binary but some of them are multi-valued such as RHit and ECat. In LSRP, if two or more corresponding logic of a specific variable are distinct, the rules with the specific variable are merged into one and then the corresponding antecedent condition of the variable will be omitted in a merged rule. Thus simpler rules can be derived. The following example with six artificial rules shows the idea well.

R1: if Age = JS and ECat = Finance and RHit = S and Adults = F and Games = T and Jobs = T then Response = Spam.

R2: if Age = JS and ECat = Finance and RHit = S and Adults = F and Games = T and Jobs = F then Response = Spam.

R3: if Age = FS and ECat = Adults and RHit = VW and Adults = T then Response = Store.

R4: if Age = FS and ECat = Adults and RHit = N and Adults = T then Response = Store.

R5: if Age = FS and ECat = Adults and RHit = S and Adults = T then Response = Store.

R6: if Age = FS and ECat = Adults and RHit = VS and Adults = T then Response = Store.

There are two similar rules R1 and R2 with only one distinct antecedent condition in Jobs attribute. Therefore the two rules are merged into  $R1 \cdot R2$  where the antecedent conditions T and F of the variable Jobs are merged into Null because Jobs = T is in conjunction with Jobs = F by logic synthesis operation. So a new rule R7 is derived by excluding the Null condition in  $R1 \cdot R2$ .

R7: if Age = JS and ECat = Finance and RHit = S and Adults = F and Games = T then Response = Spam.

Similar operation can be also performed for nominal variables. As we mentioned, a variable RHit has four categories. If RHit values of four rules being compared are distinct, then the four rules are merged into one rule. In the above example, R3 has VW value as RHit and the values of RHit in the other three rules are N, S, and VS, respectively. From the synthesis of rules 3, 4, 5, and 6, the antecedent condition of RHit should have Null and hence the condition is excluded to construct a new rule R8.

R8: if Age = FS and ECat = Adults and Adults = T then Response = Store.

It is possible that two or more rules sometimes compete to merge other rules with only one different antecedent condition. To resolve this situation, we consider information gain (IG) of each attribute in a rule set. When two or more candidates are found to be merged, we should choose a variable with lowest IG and then merge two rules with distinct binary attribute values or several rules with distinctive multiple attribute values in the variable. It is fair that the attributes with higher IGs should



survive in a rule set. As we expected, experimental results (see Section 5.3) are a little different from the ID3 mining.

Third, we interpret the derived classification rules to an ontology using a formal language, Web-PDDL [13], a strongly typed first order language especially for representing ontologies and mappings between them. Based on previous results, we first define a user preference ontology in Web-PDDL and it can be automatically translated to other popular ontology and rule languages, such as OWL [14] and SWRL [15]. We selected the following concepts as classes and properties:

Classes (Types): Preference, Event, Email, Action, Client, Gender, ECat, RHit, Response

Properties (Predicates): name, sex, age, prefer, category, respond

In Web-PDDL, it looks like

(define (domain spam\_email)

(:extends (uri "http://orlando.drc.com/daml/ontology/Person/G3/Person-ont-g3r1"  
:prefix pdt)

(uri "http://www.w3.org/2000/10/XMLSchema" :prefix xsd))

(types: Event Email Preference - Object Action - Event Client - @pdt:Person  
Gender RHit ECat Age - @xsd:string Response - Action)

(:Objects Reply Store Delete Spam - Response  
Adults Games Jobs .... - Preference)

(:predicates (name c - Client n - @xsd:string)  
(sex c - Client s - Gender)  
(age c - Client a - Age)  
(prefer c - Client e - Preference)  
(category e - Email e - ECat)  
(respond c - Client e - Email r - Response)))

Where user responses (e.g., Reply Store Delete Spam) and preferences (e.g., Adults Games Jobs) can be defined as objects (instances) of “Response” class and “Preference” class. Then the rules we got from data mining can be put into the user preference ontology as axioms. For example, R8 can be represented in Web-PDDL as axioms:

(axioms:

(forall (c - Client e - Email)  
(if (and (age c “FS”) (prefer c Adults) (category e “Adults”)  
(respond c e Store)))

In the following section, we will show how to use this ontology in an ontology-based anti-spam system.

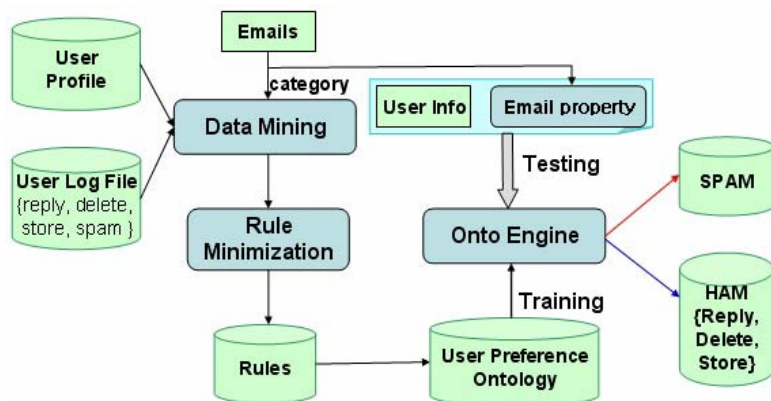
## 5 Architecture, Experiments, Results, and Discussion

In this section, we will present the architecture for our ontology-based system and several measures for evaluation of the system performance and the quality of rules derived from the LSRP process. Experimental results and some discussion are also described.

## 5.1 Architecture

Each user can respond differently to a mail with even identical mail header and content. This situation is mainly caused by personal preferences and potential modes of behaviors. However, we do not consider users' unpredictable behaviors in this paper, because the work is out of our research scope. We started from this assumption and decided to show that it was valid in real situations. Thus, we collected preferences for a group of users. To analyze potential responses of users to various emails, we provided sample emails to a user group and asked them to respond of the predefined (Reply, Delete, Store, Spam) actions. In this work, "Reply", "Delete", and "Store" responses are considered ham emails but only "Spam" response is considered as spam. Thus, our research is different from conventional anti-spam mail works because we classify user's specific responses into 4 categories instead of spam and ham.

The proposed architecture is given in Figure 2: user profile was collected from several participant users, user log file was also built with their responses to sample emails, and ECat values were given to individual mails by a human expert. We used a popular data mining tool, WEKA [11] developed by Witten and Frank to find some association and classification rules between preferences and responses. User preference ontology was constructed after data mining and rule minimization. Using the ontology, especially those axioms which we constructed, our inference engine Onto-Engine [16], which is a first order logic reasoner using generalized modus ponens, can classify the emails into four categories - Spam, Reply, Delete, and Store - based on user preferences, email category, and personal information by forward chaining.



**Fig. 2.** Architecture of the proposed ontology-based anti-spam mail system

Conventional anti-spam software provides content-oriented filtering service. However, the proposed system can give user oriented anti-spam mail service, because our system not only gives information about spam mail but also estimates user's response to an incoming mail. This approach is new and can be an essential service for email clients suffering from lots of spam messages.

## 5.2 Measures

Performance measures are required to evaluate inference results of OntoEngine using axioms in user preference ontology. There are several measures such as misclassification, accuracy, prediction, recall, and so on in anti-spam mail system field [1]. Since these measures are calculated from email contents, they can be called email-oriented measures. However, we aim at user preference oriented service and hence different measures are needed to show that the proposed user preference ontology is meaningful to the anti-spam system. In this work, we suggest three measures to achieve this goal.

First, axiom accuracy or axiom confidence is useful to calculate correctness of each axiom in the user preference ontology, which is defined as the ratio of the number of instances that match the rule only in the antecedent part to the number of instances that match the whole rule. Consider the antecedent part and conclusion of each axiom when the input attributes of a test instance exactly match the antecedent conditions of  $i$ -th axiom, we increment the match count of the axiom,  $\text{axiom}[i].\text{match}$ . If the response of the test instance is also the same as the conclusion part of the axiom, the correct count of the axiom,  $\text{axiom}[i].\text{correct}$  increments too. Then axiom confidence is calculated by dividing  $\text{axiom}[i].\text{correct}$  over  $\text{axiom}[i].\text{match}$ . The ontology with higher axiom accuracy is more preferable.

Second, conventional classification accuracy is not appropriate in this context. Consider the following scenario. Two users, Bill and John, have almost the same preferences and their responses to training instances are also very similar. They can be grouped into one small user group and their responses to any email are highly possible to be inferred as the same according to the ontology. In this situation, we also suppose that the response of the user group in which Bill and John are included has been inferred as spam for specific antecedent portion in an axiom. All possible instances, including Bill's spam response and John's delete one for the specific antecedent portion in the axiom, have been already reflected during the data mining process. After that, the system judges that Bill's response is correct and John's one is wrong for the test instance matched with corresponding antecedent portion in the axiom. It is not certain to determine which one is correct. It is not guaranteed that users' responses are equivalent to every kind of email. So we introduce axiom capacity as a measure of how many instances can be accommodated by each axiom. Therefore, if the summation of match scores of each axiom ( $\sum_i \text{axiom}[i].\text{match}$ ) is equal to the total number of test instances, then the axiom set can accommodate all instances and there is no capacity problem. However, it is not easy because we have mined several thousands of instances with binary as well as nominal attributes to derive tens of rules. We should pass outside instances away from axioms in the ontology to conventional content-based email filters such as NBC or SVM and let them process the instances.

Third, we provide a simple quantified measure to evaluate comprehensibility of a rule derived from the proposed rule minimization method (LSRP). We defined the matched term ratio in a rule,  $mt$ , for each rule as the number of attributes in each instance over the number of antecedent conditions in each rule. The greater the average value of all matched term ratio in a rule set is, the simpler and more easily interpretable the rule set is to humans. For example, a test instance, (Age = JS and ECat = Adults and RHit = S and Adults = F and Games = T and Jobs = T) and

(Response = Spam) is presented. Two rules (R9: if Age = JS and ECat = Adults and RHit = S and Adults = F and Games = T then Response = Spam) and (R10: if Age = JS and ECat = Adults and RHit = S and Adults = F then Response = Spam) are given. Then  $mt[9] = 6/5 = 1.2$  and  $mt[10] = 6/4 = 1.5$ . Therefore, the R10 rule has a greater matched term ratio than R9 in terms of quantified comprehensibility. This measure is simple and quantified. Chan and Freitas also measured rule comprehensibility by the average number of terms in the discovered rules [8] but they did not consider the number of input attributes. The above three measures help performance evaluation in terms of axiom confidence, capacity, and comprehensibility.

### 5.3 Experiments, Results and Discussion

To evaluate the proposed approach based on the user preference ontology, we collected 40 sample emails with labelled categories. We also collected responses to those emails from 90 college students together with their respective preference over several options. And thus we used a dataset with 3,600 records; each record consists of the email category, user preferences, and the corresponding response. We used 2,400 instances of the dataset in the training process, and the rest for testing by performing the rule evaluation methods described in the previous section. Before carrying out the ID3 mining, we selected 6 out of 15 attributes with the highest information gain, namely, ECat, Age, RHit, Adults, Games, and Jobs.

We got 89 rules with accuracies greater than 0% from the ID3 data mining. To evaluate the performance of derived rules, we applied them to the 1,200 test instances and carried out the proposed rule minimization approach (LSRP), which then generated a reduced amount of 77 rules. We interpreted the rules in logic axiom form and selected some representative ones where one rule was chosen for each email category, as shown in Table 1 in descending order of accuracy. Each rule in the table explains the way that a user responds to a certain email with respect to the specific preference. For example, the first axiom rule shows that 85% of users with Adults=False and RHit=Neutral preferences responded as "Spam" when they got adult-related emails. In fact, all adult emails were definitely classified to spam in content-based filters, while this experiment shows that a few users did not think of those kinds of emails as spam. This point convinces us that the proposed user preference ontology based approach can be a customized solution for individual users.

Table 2 shows a comparison of the rule set generated by using rule minimization method and the one without using it. As shown in the table, average axiom rule accuracy was degraded a little by 3.3%. However the number of axiom rules is reduced by 13.5%, and the average matched term ratio and the total capacity are also improved by 15.2% and 1.6% respectively. This reduced rule set with shorter rules is desirable to construct a user preference ontology because we pass the rules to each user and the user feedback personal preference ontology to the system by easily modifying the rule set according to his or her personal interest.

The correlation of the user's response to a certain kind of email with respect to his/her preference shown in the rules derived from the current experiment data set is not significant enough as expected. It is because the data set is not sufficient to supply samples for all pre-defined email categories and the distribution of which is unbalanced as well. And the lack of sample email data makes it hard to illustrate a clear

pattern of the user’s response to a certain email category. To increase the accuracy and practicality of the rules, more data should be collected in the future. However our work points out an innovative anti-spam approach which incorporates the user preference rather than analyzing the email content alone. The result under the current experiment setup also demonstrates a good performance of the proposed rule minimization method.

**Table 1.** Axioms derived by logic synthesis based rule pruning and their performance

No	Axiom rules	Matched term ratio	Accuracy
1	ECat=Adults $\wedge$ Adults=F $\wedge$ RHit=N $\Rightarrow$ Response=Spam	2	85.0%
2	ECat=Etc $\wedge$ Age=FS $\wedge$ Jobs=T $\wedge$ Games=T $\wedge$ RHit=S $\Rightarrow$ Response=Spam	1.2	82.4%
3	ECat=Finance $\wedge$ Age=JS $\wedge$ Adults=F $\wedge$ Games=F $\wedge$ RHit=N $\Rightarrow$ Response=Spam	1.2	81.5%
4	ECat=News $\wedge$ Age=JS $\wedge$ Jobs=F $\wedge$ Adults=F $\wedge$ RHit=N $\Rightarrow$ Response=Delete	1.2	75.0%
5	ECat=TVMovieMusic $\wedge$ Age=JS $\wedge$ Jobs=T $\wedge$ RHit=N $\Rightarrow$ Response=Reply	1.5	71.4%
6	ECat=IT $\wedge$ Age=JS $\wedge$ Jobs=T $\wedge$ Games=T $\wedge$ RHit=N $\Rightarrow$ Response=Delete	1.2	65.9%
7	ECat=Shopping $\wedge$ Age=JS $\wedge$ Adults=F $\wedge$ Games=F $\wedge$ RHit=N $\Rightarrow$ Response=Spam	1.2	50.0%
8	ECat=Travel $\wedge$ Age=JS $\wedge$ Jobs=T $\wedge$ Games=F $\wedge$ RHit=N $\Rightarrow$ Response=Store	1.2	45. 5%
9	ECat=Jobs $\wedge$ Age=JS $\wedge$ Jobs=T $\wedge$ Adults=F $\wedge$ RHit=N $\Rightarrow$ Response=Spam	1.2	34.6%

**Table 2.** Comparison on experimental results for two axiom rule sets derived by the original ID3 mining and the proposed rule pruning

Method	Number of axiom rules				Axiom accuracy	Capacity	Matched term ratio
	Reply	Delete	Store	Spam			
ID3	10	18	17	44	60.1%	1111/1200	1.25
LSRP	8	16	15	38	58.1%	1129/1200	1.44
Improv.	20%	11.1%	11.8%	13.6%	-3.3%	1.6%	15.2%

## 6 Conclusion

We proposed a method to construct user preference ontology for anti-spam mail systems. The important feature of our approach is to allow users to give different response to the same email based on their preferences. It is different from conventional systems that normally judge which mail is spam based on the email content and expect every user to equally respond to the same email. It is a big step forward

personalized anti-spam mail service considering user preference and previous response history as well as email content. The most important contribution of this work is that a user preference ontology can explain why a mail is decided to be spam or ham in a meaningful way. Also, the logic rules found by data mining contribute to this purpose and a rule pruning method improves human comprehensibility. For the future work, we need to extend the proposed system to process real-time and larger number of emails to compare our system's performance with conventional content-based filters. We expect that users have consistent responses when the volume of experiment corpus is bigger and thus the testing result can reflect the impact of the preferences over their decision.

**Acknowledgement.** The first author was supported by the Korea Research Foundation Grant. (KRF-2006-013-D00285) He has worked as a visiting scholar at the AIM Lab during his sabbatical year 2006 and thanks to the Department of Computer and Information Science at the University of Oregon. Also we appreciate 90 participant students to give their preferences and feedback their responses to sample emails.

## References

1. Cormack, G. V., Overview of the TREC 2005 Spam Track, <http://plg.uwaterloo.ca/~gvcormac/trecspamtrack05>
2. Wolfe, P., Scott, C., and Erwin, M., Anti-Spam Tool Kit, McGraw Hill (2004)
3. Gray, A. and Haahr, M., "Personalized, Collaborative Spam Filtering," in Proc. of the First Conference on Email and Anti-Spam (2004)
4. Ravi, J., Shi, W., and Xu, C., "Personalized Email Management at Network Edges," IEEE Internet Computing, Vol.9(2) (2005) 54-60
5. Anti-Spam Firewall, [http://www.barracudanetworks.com/ns/products/anti\\_spam\\_tech.php](http://www.barracudanetworks.com/ns/products/anti_spam_tech.php)
6. Maedche, A., "Ontology Learning for the Semantic Web," The Kluwer International Series in Engineering and Computer Science, Volume 665 (2003)
7. Files, C. M. and Perkowski, M. A., "Multi-Valued Functional Decomposition as a Machine Learning Method," in Proc. of ISMVL '98 (1998) 173-178
8. Chan, A. and Freitas, A., "A New Classification-Rule Pruning Procedure for an Ant Colony Algorithm," LNCS 3871 (2005) 25-36
9. Sasao T., "Switching Theory for Logic Synthesis," Kluwer Academic Publishers (1999)
10. Kim, J. and Kang, S., "Feature Selection by Fuzzy Inference and Its Application to Spam-Mail Filtering," LNAI 3801 (2005) 361-366
11. Witten, I. H. and Frank, E., Data Mining: practical machine learning tools and techniques, 2<sup>nd</sup> ed, Morgan Kaufmann (2005)
12. Gruber, T. R., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," Int. Journal of Human-Computer Studies, Vol.43 (1995) 907-928
13. McDermott, D. and Dou D., "Representing disjunction and quantifiers in RDF," in Proc. Int'l Semantic Web Conference (2002) 250-263
14. OWL Web Ontology Language. <http://www.w3.org/TR/owl-ref/>
15. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. <http://www.w3.org/Submission/SWRL/>
16. Dou, D., McDermott, V., and Qi, P., "Ontology translation on the semantic web," Journal of Data Semantics, Vol.2 (2004) 35-57

# Question Answering Summarization of Multiple Biomedical Documents

Zhongmin Shi, Gabor Melli, Yang Wang, Yudong Liu, Baohua Gu, Mehdi M. Kashani, Anoop Sarkar, and Fred Popowich

School of Computing Science, Simon Fraser University  
Burnaby, BC V5A 1S6, Canada

**Abstract.** In this paper we introduce a system that automatically summarizes multiple biomedical documents relevant to a question. The system extracts biomedical and general concepts by utilizing concept-level knowledge from domain-specific and domain-independent sources. Semantic role labeling, semantic subgraph-based sentence selection and automatic post-editing are involved in the process of finding the information need. Due to the absence of expert-written summaries of biomedical documents, we propose an approximate evaluation by taking MEDLINE abstracts as expert-written summaries. Evaluation results indicate that our system does help in answering questions and the automatically generated summaries are comparable to abstracts of biomedical articles, as evaluated using the ROUGE measure.

## 1 Introduction

With the rapid development of biological and medical research in the last decade, the volume of biomedical scientific articles has greatly increased. For instance, over 2,000 new articles are being added to the MEDLINE database every day. It is extremely difficult for physicians and researchers in medicine and biology to build up their own knowledge base from existing publications and update it daily. Therefore automatic methods such as summarization and question answering (QA) that can quickly understand and find the main points of biomedical articles are becoming more essential.

Domain-independent summarizers, such as WebSumm [1], Newsblaster<sup>1</sup> and Alias-I<sup>2</sup>, have been used to generate summaries of biomedical articles [2]. However, when tuning a summarizer to a particular domain, domain specific information would improve the quality of summaries. Gaizauskas et al. proposed TRESTLE (Text Retrieval Extraction and Summarization Technologies for Large Enterprises) that relies on named entity annotations and scenario templates to generate single sentence summaries of pharmaceutical news archives [3]. Centrifuser [4] is a summarization system that computes topical similarities among documents using a tree structure-based calculation and extracts sentences from topic-relevant

---

<sup>1</sup> <http://www.cs.columbia.edu/nlp/newsblaster/>

<sup>2</sup> <http://www.alias-i.com/>

documents. Elhadad and McKeown introduced a summarizer that generates a patient-specific summary from journal medical articles [5]. In this system documents are first categorized into main clinical tasks, from which a set of templates are built and matched with patient records. Patient-relevant templates are then merged and ordered to generate fluent English text.

In this paper we introduce BIOSQUASH, a question-oriented extractive summarization system on biomedical multi-documents that are relevant to a question. The system was based upon a general-purpose summarizer, SQUASH [6]. We propose a method to utilize concept-level characteristics from a domain-specific ontology, UMLS (Unified Medical Language System<sup>3</sup>), and a domain-independent lexicial reference system, WordNet<sup>4</sup>. Details of the system design are described in the rest of this paper as follows. §2 provides a high-level description of the BIOSQUASH system. The automatic annotation of the input documents to be summarized is described in §3 and §4. Construction of the semantic graph and the sentence extraction step based upon concept-level characteristics are discussed in §5. §6 introduces our redundancy elimination and sentence ordering strategies to produce more readable summaries. The evaluation on experimental results is given in §7. Some discussions and future work are brought forward in §8.

## 2 The System Architecture

The BIOSQUASH system has four main components: the Annotator, Concept Similarity, Extractor and Editor modules, as illustrated in Fig. 1. The system starts off by annotating the documents and the question text with syntactic and shallow semantic information in the **Annotator module**. These annotations do not provide sufficient semantic background when we study the relations among concepts in documents and questions. The actual semantic meanings of both general and biomedical concepts as well as the ontological relations among these concepts are obtained in the **Concept Similarity Module**.

The annotations and conceptual information are then fed to two summarization stages: the first is the **Extractor module**, which focuses on content selection and aims to optimize the ROUGE<sup>5</sup> score; while the next stage, the **Editor module**, focuses on linguistic readability. In the Extractor module, a semantic graph is constructed based on the semantic role labeling and the conceptual information of documents as well as the question text. Sentence selection is performed by sub-graph selection on the semantic graph. Sentence redundancy is also measured and used to create sentence clusters related to the topic question. The Editor module orders sentences from the sentence clusters provided by the Extractor, eliminates irrelevant content from long sentences and finally produces the summary conforming to the length limit.

---

<sup>3</sup> <http://www.nlm.nih.gov/research/umls/>

<sup>4</sup> <http://wordnet.princeton.edu/>

<sup>5</sup> Recall-Oriented Understudy for Gisting Evaluation, <http://haydn.isi.edu/ROUGE/>



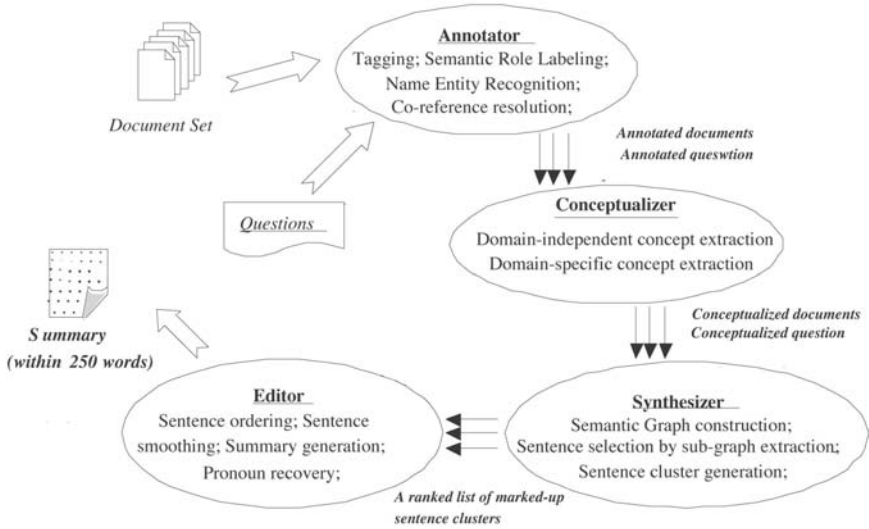


Fig. 1. The overall system design of BIOSQUASH

### 3 The Annotator Module

The annotations used in the BIOSQUASH system include output of a statistical parser [7], a named-entity recognizer and a semantic role labeler. The named-entity recognition (NER) sub-module categorizes atomic text elements into predefined named entity classes. We use Alias-I's Lingpipe system to identify persons, organizations, locations, numeric entities, pronoun entities as well as biomedical entities as defined in the GENIA ontology<sup>6</sup>.

A semantic role is the relationship that a syntactic constituent has with a predicate. Typical semantic roles for arguments of the predicate include Agent, Patient, Instrument, and semantic roles for adjuncts include indicating Locative, Temporal, Manner, Cause, among others. The task of semantic role labeling is, for each predicate in a sentence, to identify all constituents that fill a semantic role and to determine their roles, if any [8,9]. Recognizing and labeling semantic arguments is a key task for answering “Wh-” and other more general types of questions in the summarization task. Automatic semantic role labeling methods have been discussed in depth in [8]. The semantic role labeling (SRL) for documents and questions is produced by transducing the output of the statistical parser using our own SRL system [10], which is trained on the semantic annotations provided by the CoNLL-2005 data-set, a modified version of the annotation provided by the Penn PropBank data-set [11]. The following example illustrates the input and output of the SRL sub-module:

<sup>6</sup> <http://www-tsujii.is.s.u-tokyo.ac.jp/genia/topics/Corpus/genia-ontology.html>

### Example 1

*Input:* (S (NP (NP (DT The) (NN processing)) (PP (IN of) (NP (NNP LcnA)))) (VP (VBZ involves) (NP (NNP LcnC))) (. .))

*Output:* (S (<sub>A0</sub> (NP (NP (DT The) (NN processing)) (PP (IN of) (NP (NNP LcnA)))) (VP (VBZ *PREDICATE* involves) (<sub>A1</sub> (NP (NNP LcnC))) (. .))<sup>7</sup>

## 4 The Concept Similarity Module

Identification of similarity between questions and sentences in documents is crucial to our QA summarization task, in which sentences in documents are selected based on their similarities with other sentences in documents and questions. Specifically, the sentence similarity metric is used to choose the significant “group” of sentences and to decide the relevance of a sentence to the question. A sentence similarity confined to word surface patterns and simple string matching would however fail in cases of:

- **identifying synonyms.** Synonyms with different lexical forms are not taken into account in the sentence similarity metric using only the word-based approach. This problem is especially sensitive in biomedical documents, since many biological and medical substance names have various lexical forms. For instance, when the question asks “the role of the gene BARD1 in the process of BRCA1 regulation”, we would expect terms like “function”/“character”, “BARD 1”, “Breast Cancer 1 Protein” to be considered as similar.
- **identifying hypernym-hyponym relations.** If the question contains hypernyms of words in the sentence, the word-based approach does not consider the sentence to be similar to the question. For instance, the occurrence of the words “arbovirus”, “bacteriophage” and “viroid” in the sentence should improve the sentence significance when the question involves their hypernym, “virus”.
- **word sense disambiguation.** The word-based approach would take lexically identical words as the same, even though they occur different senses in a particular context.

We define **Super Concept**, as a synonym, hypernym (*is-a*) or holonym (*part-of*) of a concept. Therefore, super concept is reflexive and transitive: 1) Any concept is a super concept of itself; 2) If concept *A* is a super concept of *B* and *B* is a super concept of *C*, then *A* is a super concept of *C*. We say two concepts are related ontologically if one is the super concept of the other.

The BIOSQUASH system recognizes each concept by a **Concept ID** (CID), a unique identification of each distinct entity, event and relation. Concepts with the same CID are synonyms. In addition, hypernyms of each concept are also provided, therefore the system would ideally not have above three problems after the conceptualization.

<sup>7</sup> We use the Role Set defined in the *PropBank Frames scheme* [9].

Concepts and their hypernyms are extracted from two public ontologies: WordNet for domain-independent concepts and UMLS for domain-specific concepts. We apply a CPAN module<sup>8</sup>, WordNet::SenseRelate::AllWords, to select the correct sense of each word. The module is an implementation of a word sense disambiguation algorithm that measures similarity and relatedness based on the context of the word and the word sense glosses in WordNet [12]. Table 1 shows an example of sentence annotation that can allow matching according to concept similarity by matching CIDs.

**Table 1.** An example of a sentence annotated with concept ID’s to allow matching according to concept similarity

WORD	The	processing	of	LacZ	involves	LenC	.
WordNet_CID	-	(13366961)	-	-	(02602586)	-	-
UMLS_CID	-	-	-	(C0022959)	-	(C1448241)	-

## 5 The Extractor Module

The Extractor takes the results from the Annotator and Concept Similarity Modules (see §3 & §4) and provides relevant sentences to the Editor module (see §6). The extractor module performs the following tasks.

### 5.1 Concept Identification

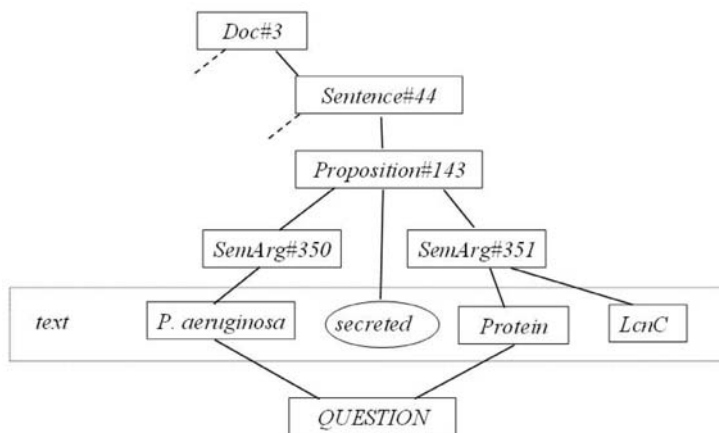
The first task of the Extractor is to locate the concepts that exist in the document set. Given the deep syntactic and shallow semantic annotation, a relatively complete set of concepts is extracted. Three types of concepts are located: ontological concepts, named entities and noun phrases. An ontological concept is a phrase that Concept Similarity Module has connected to one of the available ontologies (WordNet or UMLS). For example, the well-studied organism *Pseudomonas aeruginosa*, the phrase *nucleotide sequence*, and the word *located* would likely be assigned a concept because the Concept Similarity Module would recognize it as a member of UMLS. Similarly, a named entity concept is a text phrase that Annotator has identified to be a certain named entity type. For example, the recently discovered protein *macrophage inflammatory protein-1a* could be identified by a named-entity recognizer (NER) but not by Concept Similarity Module because the ontology is dated and/or curated. Finally, a word that is not recognized as either a named entity or a member of the available ontologies, but is recognized to be a noun phrases by the statistical parser, is also labelled as a concept.

### 5.2 Text Graph Creation

Once the concepts have been identified within the documents, the next task performed by the Extractor is to identify linguistic and semantic relationships

<sup>8</sup> <http://www.cpan.org/>

between them. The relationships between concepts result in graph edges. More specifically, the semantic relations of a document can be given by the semantic labeler. As in [13] a *semantic graph* is used as a semantic representation of the document. Because the system works on multiple documents for one question, the semantic graph represents the semantic relations for all the documents by sharing the common nodes and links. An example of a small portion of the semantic graph constructed from a set of documents is shown in Figure 2. Constructed from the output of the Annotator, the text graph contains the information in the text of the documents and question that is essential to topic-based multi-document summarization.



**Fig. 2.** Subgraph of the overall text graph associated with the proposition *P. aeruginosa secreted the protein LcnC*. The dashed lines indicate multiple edges that likely exist but are not included in the figure. In this example the question asks for information about proteins associated with *P. aeruginosa*.

### 5.3 Concept and Proposition Significance

Given a text graph, each concept is assigned a **significance** score and then each proposition is scored based on concepts it contains. The value of this score is based on 1) the number of edges to questions, documents and propositions; 2) whether the type is an ontological concept, named entity or noun phrase. Next, each proposition in every sentence from every document is given a significance score, which will be used to rank the relevance of each proposition to the summary. The contribution to the significance score is calculated as the summation of the individual significance values for each unique entity in the sentence. Further details about assigning the significance score are described in [6].

### 5.4 Covering the Concept Space

The selection of propositions occurs sequentially, and a fixed number of sentences is returned. This number is large enough to result in more sentences than

strictly needed for the summary length limit. The selection of the first proposition is simply based on the largest significance score, with ties broken randomly. Once the first proposition is selected, similar propositions are of less value to Extractor. These similar propositions are penalized in order to ensure that other interesting topics were selected. This process is iterated until the required number of sentences was selected. Further details about the penalty function are introduced in [6].

## 6 The Editor Module

The task of the Editor module is to produce a fluent summary. To achieve this, we order all the sentences based on their significance scores produced by the Extractor module, and select the highest scoring subset of the sentences as the candidate sentences for the summary. Those sentences are then re-ordered by a 2-phase sentence ordering algorithm and a summary candidate is generated after compressing the re-ordered sentences. The sentence compression step deletes words or phrases that involve the use of discourse or chronological markers that can affect fluency of the summary after re-ordering.

We propose a two-phase ordering algorithm to assign an **Importance** score to each sentence and order them. In the first phase, the importance score  $g$  of each sentence  $s_i$  is computed as a linear combination of a list of features:  $g(s_i) = w_1 F_1 + \dots + w_n F_n$ , where  $F_j$  is the value of the  $j$ th feature equal to either 0 or 1.  $w_j$  is the corresponding interpolation weight of  $F_j$ . We manually set  $w_j$  based on a study of existing summaries. The following features are used to calculate the importance score: information importance from Extractor module, question and sentence overlap, first and last sentences in the document and sentence-length cutoff.

The sentences are then re-ordered by their importance scores. In order to choose the sentences that are coherent to their neighbors in the summary, we calculate the similarity score of two sentences based on their Longest Common Subsequences (LCS), as the second phase of ordering. More specifically, to choose the  $k$ th sentence  $m_k$  of the summary, each of the rest sentences is measured against the sum of two scores in said ordering phases:  $m_k = \operatorname{argmax}_{s \in S'} g(s) + LCS(s, m_{k-1})$ , where  $S'$  denotes the set of sentences that have not been selected in the summary. Note that, although we have not experimented with different sentence similarity measures, LCS may be replaced by other techniques, for instance, edit distance and n-gram comparison.

## 7 Experiments and Evaluations

### 7.1 Data

One of the challenges to developing a document summarizer for biomedical documents is measuring the quality of the machine-generated summaries. The ideal scenario is to have the summaries evaluated by experts in biomedicine. Unfortunately, as in most research settings, we did not have access to a pool of experts to evaluate our summaries. The more typical method of evaluation is to use an

n-gram based algorithm to compare the machine-generated summaries to a set of human written summaries (for which we also need domain experts to produce multiple summaries for the same question/topic). The annual Document Understanding Conference (DUC)<sup>9</sup> for example hires humans to write summaries of newspaper articles that answer a specific questions, such as: *What countries have chronic potable water shortages and why?* To the best of our knowledge however, no dataset has been explicitly created to test automated summarization from the biomedicine domain.

We solve this problem by transforming data from the Ad-hoc Retrieval task of Genomics Track at the 2005 Text Retrieval Conference (TREC)<sup>10</sup> in order to test the BIOSQUASH system. The original TREC task was: given a question on relationships that exist among biological substances and/or processes, retrieve a set of documents that are relevant to the question from a 4.5-million document subset of the MEDLINE database. To transform this dataset to our needs we simply treat the paper abstracts as substitutes for expert-written summaries. Our assumption is that a paper abstract selected to be relevant by a human judge will closely approximate the summarized answer sought from the question. Figure 3 contains a sample question from the TREC task.

From the TREC data we selected the 18 questions (the precise question IDs are provided in Fig. 4) with the most number of documents associated with it. The questions with a large number of documents were required in order to effectively apply ROUGE, a recall oriented n-gram matching measure of summarization quality [15]. For each TREC question the relevant abstracts are separated into three subsets: 5 abstracts as peers (each peer abstract is taken to be a human written summary and compared our machine generated summary), 30 abstracts that will be summarized by BIOSQUASH and 30 abstracts to be used as references by ROUGE.

A fourth set of abstracts was associated to each of the 18 selected questions. This set is based on the documents retrieved by one of the Information Retrieval (IR) systems that participated in the TREC task [14]<sup>11</sup>. This set allows us to present results for a real-life system that both retrieves appropriate documents and produces a summary from them. Figure 3 presents one of selected questions and the corresponding summary generated by BIOSQUASH from documents retrieved by the chosen IR system.

In summary, the following sets of abstracts are available for each of the 18 questions:

- **Peer Abstracts:** Five of the human selected abstracts are treated as independently written human expert summaries.
- **Relevant Abstracts:** 30 of the human selected abstracts are used to create a summary with BioSquash. This summary is supports the scenario in which documents are manually selected by a human expert and the task is to automatically summarize the documents.

<sup>9</sup> <http://www-nlpir.nist.gov/projects/duc/>

<sup>10</sup> <http://ir.ohsu.edu/genomics/>

<sup>11</sup> this system's performance was within the average of all submitted systems (Means of Average Precision: 0.1834).

**Question:** *Provide information about the role of the gene Apolipoprotein E (ApoE) in the disease Alzheimer's Disease.*

**Summary:** *The raised frequency of the epsilon 4 allele in the patients with onset Alzheimer's disease was of a lower magnitude than that in United States and Canadian studies. The frequency of E4 alleles was increased ( $\chi^2 = 42$ ;  $df = 1$ ;  $p < 10$ ) among patients with Alzheimer's disease compared with a Danish control population. Apolipoprotein E is associated with Alzheimer's disease neurofibrillary tangles and beta-amyloid protein in senile plaques. The relative epsilon 4 allele frequency was 0.472 in LBD, 0.513 in AD-CVD, 0.405 in presenile AD, 0.364 in senile AD, and 0.079 in vascular dementia. To clarify the association of ApoE polymorphism with Alzheimer's disease and vascular dementia in Japan, 13 patients with early onset sporadic Alzheimer's disease, 40 patients with late onset sporadic Alzheimer's disease, 19 patients with vascular dementia, and 49 non-demented control subjects were analysed. Apolipoprotein E sigma4 allele is associated with Alzheimer's disease in familial and sporadic cases, but the associations of ApoE sigma4 allele and vascular dementia and/or ischemic cerebrovascular disease are still controversial. Alzheimer's disease is associated with an increased frequency of the apolipoprotein E type epsilon 4 allele. The epsilon 4 allele has also been shown to reduce the age at onset of dementia in AD in a dose dependent manner, with the epsilon 2 allele having an opposing effect. Apolipoprotein E epsilon 4 allele frequency among Alzheimer's disease patients is increased compared to control subjects and is influenced by the presence of other genetic factors and age at symptom onset.*

**Fig. 3.** One of the TREC questions (q117) and the corresponding summary generated by BIOSQUASH from documents retrieved by the IR system [14]

- **Retrieved Abstracts:** 30 of the system retrieved abstracts are used to create a summary with BioSquash. This summary supports the scenario in which the human expert only provides the question and both the retrieval and summarization are automatically performed.
- **Reference Abstracts:** Up to 50 of the human selected abstracts are used as the full set of gold standard summaries when computing the ROUGE score for the peer “summary” or machine generated summary.

## 7.2 Evaluation Measures and Results

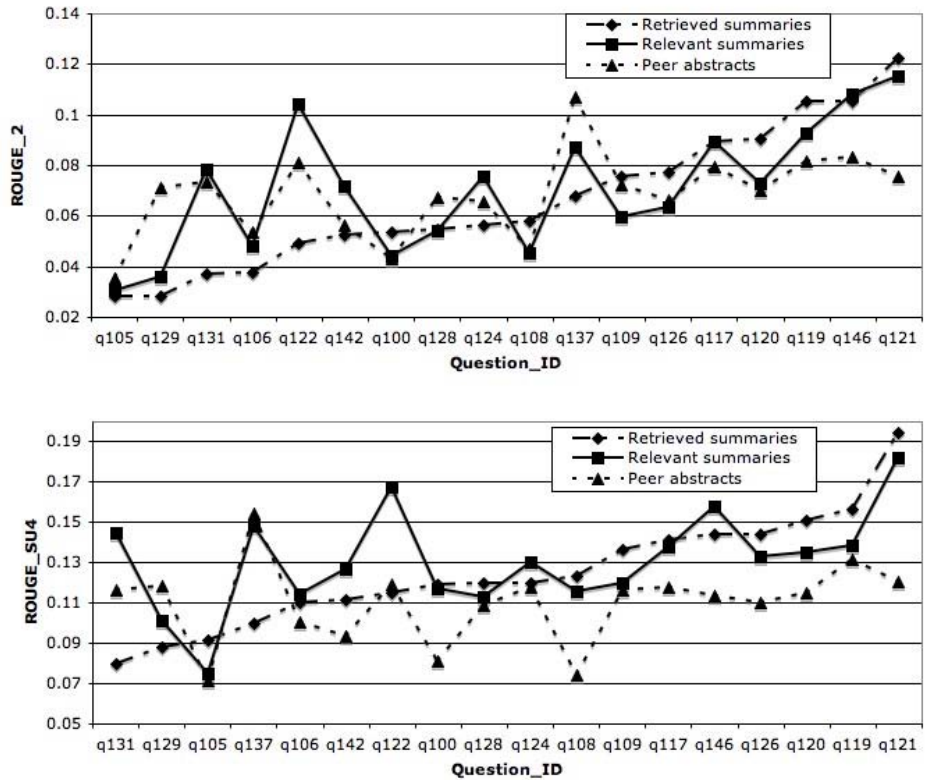
We follow the evaluation methodology of DUC 2005 and DUC 2006 that used ROUGE-2 and ROUGE-SU4 to measure performance. ROUGE-n is an n-gram recall-oriented measure to compare a candidate summary with a set of reference summaries [15]. ROUGE-SU4 is similar to ROUGE-n except that it involves bi-grams with maximum skip distance of 4 [15].

As described in §7.1, our experiment's ROUGE scores measure three different sets of candidate summaries (peer abstracts, summaries of relevant abstracts and summaries of retrieved abstracts) against the reference abstracts. The evaluation of summaries based on retrieved abstracts is related to the setting which an IR module is used to retrieve several documents relevant to the query, and then our system is used to produce a question-focused summarization. Table 2 shows ROUGE-2

and -SU4 scores for the three sets, among which summaries of relevant abstracts outperform others on both ROUGE-2 and -SU4. The results indicate that our question-focused machine generated summaries contain information that is more pertinent to the question when compared to human-written abstracts (which were not written with any particular question in mind, but rather summarize the document for which the abstract was written). Figure 4 illustrates ROUGE-2 and -SU4 scores of the three sets of candidate summaries for all chosen questions.

**Table 2.** ROUGE-2 and -SU4 scores of the three sets of candidate summaries. For peer abstracts, scores listed are the average of ROUGE scores of 5 peer abstracts.

Candidate Summary	ROUGE-2	ROUGE-SU4
Summaries of relevant abstracts	<b>0.0697</b>	<b>0.1300</b>
Summaries of retrieved abstracts	0.0669	0.1248
Peer abstracts (average)	0.0690	0.1118



**Fig. 4.** ROUGE-2 (upper) and -SU4 (lower) of three candidate summary sets for all selected questions, sorted by the performance of BioSquash on abstracts that were retrived by the IR system



## 8 Conclusion and Future Work

Text summarization in a professional domain, e.g., biology and medicine, is a very challenging task due to the need of domain knowledge in understanding domain-specific contents and a great amount of co-operation from domain experts, especially with respect to evaluation of summarization systems. In this paper we proposed a QA-based summarization system, BIOSQUASH, that automatically produces a summary of biomedical multi-documents relevant to a question. The system utilizes conceptual information extracted from both domain-independent and domain-specific ontologies to create fluent 250-word summaries relevant to a user question.

Due to the absence of expert-written biomedical summaries, in the evaluation phase we instead treat MEDLINE abstracts as expert-written summaries when calculating ROUGE scores. We do not know how close this approximate evaluation would be to the ideal evaluation procedure before a certain amount of expert-written summaries is available. However, our experimental results indicate that: 1) machine-generated summaries of abstracts known to be relevant to the question have better quality in terms of n-gram matching than the human written abstracts that are oblivious to the question; 2) machine-generated summaries of abstracts that were retrieved using an IR system with the question as the query are comparable to abstracts known to be relevant to the question.

The evaluation involves no linguistic readability of summaries, since the work needs significant amount of expert participation that is not available to our system at this stage. We are planning a more comprehensive evaluation on the BIOSQUASH system by collecting human sources, e.g., expert-written summaries and domain expert assessors in future work. We plan to evaluate our system on full articles in addition to abstracts.

## References

1. Mani, I., Bloedorn, E.: Summarizing similarities and differences among related documents. *Information Retrieval* **1**(1) (1999) 1–23
2. Damianos, L., Day, D., Hirschman, L., Kozierok, R., Mardis, S., McEntee, T., McHenry, C., Miller, K., Ponte, J., Reeder, F., van Guilder, L., Wellner, B., Wilson, G., Wohlever, S.: Real users, real data, real problems: the mitap system for monitoring bio events. In: *Proceedings of BTR2002*, The University of New Mexico (March 2004)
3. Gaizauskas, R., Herring, P., Oakes, M., Beaulieu, M., Willett, P., Fowkes, H., Jonsson, A.: Intelligent access to text: integrating information extraction technology into text browsers. In: *Proceedings of HLT 2001*, San Diego (2001) 189–193
4. Kan, M., McKeown, K., Klavans, J.: Domain-specific informative and indicative summarization for information retrieval. In: *Workshop on text summarization (DUC 2001)*, New Orleans (2001)
5. Elhadad, N., McKeown, K.: Towards generating patient specific summaries of medical articles. In: *Proceedings of automatic summarization workshop (NAACL 2001)*, Pittsburgh, PA, USA (2001)

6. Melli, G., Wang, Y., Liu, Y., Kashani, M., Shi, Z., Gu, B., Sarkar, A., Popowich, F.: Description of squash, the sfu question answering summary handler for the duc-2005 summarization task. In: Proceeding of DUC-2005, Vancouver, Canada (October 2005) 103–110
7. Charniak, E.: A maximum-entropy-inspired parser. In: Meeting of the North American Chapter of the ACL. (2000) 132–139
8. Gildea, D., Jurafsky, D.: Automatic labeling of semantic roles. *Computational Linguistics* **28**(3) (2002) 245–288
9. Palmer, M., Gildea, D., Kingsbury, P.: The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* **31**(1) (2005)
10. Liu, Y., Sarkar, A.: Using ltag-based features for semantic role labeling. In: Proceedings of the Eighth Workshop on Tree Adjoining Grammars and Related Formalisms: TAG+8, Poster Track, Sydney, Australia (July 2006)
11. Kingsbury, P., Palmer, M.: From treebank to propbank. In: In Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002). (2002)
12. Pedersen, T., Banerjee, S., Patwardhan, S.: Maximizing semantic relatedness to perform word sense disambiguation. In: University of Minnesota Supercomputing Institute Research Report. (March 2005)
13. Mani, I., Bloedorn, E.: Multi-document summarization by graph search and matching. In: Proceedings of the 14th National Conference on Artificial Intelligence, Providence, Rhode Island (1997) 622–628
14. Shi, Z., Gu, B., Popowich, F., Sarkar, A.: Synonym-based query expansion and boosting-based re-ranking: A two-phase approach for genomic information retrieval. In: the Fourteenth Text REtrieval Conference (TREC 2005), NIST, Gaithersburg, MD. (October 2005)
15. Lin, C.Y., Hovy, E.H.: Automatic evaluation of summaries using n-gram co-occurrence statistics. In: Proceedings of Language Technology Conference (HLT-NAACL 2003), Edmonton, Canada (May 2003)

# A Profit-Based Business Model for Evaluating Rule Interestingness

Yaohua Chen, Yan Zhao, and Yiyu Yao

Department of Computer Science, University of Regina  
Regina, Saskatchewan, Canada S4S 0A2  
{chen115y,yanzhao,yyao}@cs.uregina.ca

**Abstract.** Different types of rules are mined from transaction databases often with the goal of improving sales and services. In this paper, we link the interestingness of rules with the context of business marketing. We consider the profits generated from some specific marketing strategies that are developed based on particular discovered rules. This leads to a profit-based business model for evaluating rule interestingness. With this additional utility, we investigate some relationships between different marketing strategies and fundamental properties of rules for profit increasing.

## 1 Introduction

The rule mining was initiated as “market basket analysis,” which is used to discover co-occurrence or correlation relationships in a set of commercial items that are recorded in a large volumes of customer transaction database [1]. For example, an association rule can be expressed as “for people who buy spaghetti, wine, and sauce ( $A$ ), they also buy garlic bread ( $B$ ),” or symbolically,  $A \Rightarrow B$ . Since rule mining is virtually practical, it has a direct impact on business and takes up many actionable concerns such as maximizing profit and payoff, minimizing cost, and finally leading to a wise action and decision making [2].

From the application view of rule mining, a rule is considered to be interesting if it is novel, potentially useful, understandable, actionable, profitable or explainable [4,8]. Moreover, a rule is only advantaged while it can be understood and rationalized [19]. These lead to different philosophies in designing data mining solutions to real world problems and measures to evaluate rule interestingness [7]. Based on measurement and utility theories, a measure is proposed to evaluate one aspect of usefulness or interestingness of rules with respect to a particular context or user preference [5,15,18]. Different preferences of discovered rules must be represented quantitatively by different measures. It is difficult to justify and argue the usefulness or interestingness of a rule without concerning its usage or preference [17]. One needs to examine various circumstances in which rules are built and applied. Similar to the practice of medicine, the medicine treatment plans are determined based on the medical knowledge and practically clinical analysis and judgements [3].

In this paper, we link conventional rule mining with the concern of business profit. Typically, a discovered rule can be viewed as a type of knowledge or a belief about

purchasing behaviors or patterns of customers. Thus, rules can be used to design promotional packages or arrange cross-selling aiming to achieve more profit. For example, suppose a department store attempts to use an association rule,  $A \Rightarrow B$ , to improve sales and services [1,16]. Items in  $A$  and  $B$  may be located adjacent to each other in order to achieve an implicit recommendation and provide customers more convenience based on the learned rule  $A \Rightarrow B$ .

A simple way to evaluate rules is to apply the marketing strategies developed from rules to the real world markets, and then check whether higher profits can be generated. In fact, a profit-based model is necessary and essential to evaluate the interestingness of discovered rules. However, there exist some difficulties to implement such strategies in practice. Therefore, an alternative way is to build a model to analyze the discovered rules based on some basic business assumptions. This offers us a new interpretation and view of rules.

In this paper, we introduce a profit-based business model and consider two types of marketing strategies to increase profits. The relationships among discovered rules, marketing strategies and profit increments are investigated. In other words, the statistical factors of rules are analyzed based on the criterion, such that the corresponding marketing strategies can generate high profits. The results show that different types of rules might be useful for different types of marketing strategies.

The study proposed in this paper provides a basic and novel approach to evaluate the interestingness of rules in the context of business and economy. It is evident that an evaluation or measurement of rules should be linked to some particular contexts. This result may provide new opportunities and challenges on research and development of data mining systems.

## 2 Related Work

Many economic utilities have been studied in order to identify cost-effective and profit-benefit association rules [12,14,16,17]. Kleinberg *et al.* provided a microeconomic view of data mining [10]. They considered decision theory in the domain of data mining and argued that data mining is about extracting actionable rules which can increase utility, such as profit, security or loyalty. Wang *et al.* proposed a profit-based association rule mining model, called profit mining. They focused on a recommender that recommends items that maximize profit from future customers [16]. Lin *et al.* argued that people should consider the use of rules in the context of marketing [11]. An added value, such as profit, privacy, importance, uncertainty, or benefit of itemsets, is introduced into association rule mining model.

Some measures in actionable rule mining deals with profit-driven actions required by business decision making [12,13,14]. A rule is referred to as actionable if the user can apply it to do something. An action may be some business strategies or promotions to change the non-desirable/non-profitable rules to desirable/profitable rules. For business users, actionable rule mining can help them to influence and control their changes or actions to obtain higher profits from consumers.

### 3 Rules and Probabilistic Interpretations

In data mining, rules are typically interpreted in terms of probability. Different probabilistic measures can be defined to reflect various aspects of rules [20]. In this section, several probabilistic measures are investigated.

The sales data of a supermarket or a company can be recorded as a binary table of transactions, called a transaction table. A transaction table can be formally defined by

$$S = (T, I, \{V_i \mid i \in I\}, \{R_i \mid i \in I\}),$$

where

$T$  is a finite nonempty set of transactions,

$I$  is a finite nonempty set of items,

$V_i$  is a set of binary values  $\{0, 1\}$  for  $i \in I$ ,

$R_i : T \rightarrow V_i$  is a function between transactions and items.

Each function  $R_i$  maps a transaction in  $T$  to a binary value of  $V_i$  for an item  $i \in I$ . If a transaction  $t \in T$  contains an item  $i$ , then the value  $R_i(t)$  is 1, otherwise,  $R_i(t)$  is 0. In a transaction table, the rows correspond to transaction records, the column correspond to items, and each cell is a binary value of a transaction with respect to an item.

A subset of items  $A \subseteq I$  is called an itemset. Each transaction  $t$  contains an itemset. For an itemset  $A \subseteq I$ , let  $m(A)$  denote the portion of database formed by transactions containing all items in  $A$ , namely,

$$m(A) = \{t \in T \mid \forall i \in A, R_i(t) = 1\}.$$

The itemset  $A$  acts as a condition for the selection of transactions from the transaction database  $T$ . With this notation, we have  $T = m(\emptyset)$ .

For two disjoint itemsets,  $A, B \subseteq I$  and  $A \cap B = \emptyset$ , an implication of the form  $A \Rightarrow B$  is used to express a rule. It shows the relationship between purchasing itemset  $A$  and purchasing itemset  $B$ . A quantitative measure, called *generality*, of an itemset  $A \subseteq I$  is defined by:

$$G(A) = \frac{|m(A)|}{|T|} = P(A),$$

where  $|\cdot|$  denotes the cardinality of a set, and  $P$  denotes the probability in statistics. This measure indicates the relative size of the transactions containing the itemset  $A$ . The quantity may be viewed as the probability of a transaction  $t$  containing itemset  $A$ . Obviously, we have  $0 \leq G(A) \leq 1$ . Moreover, the generality of the itemsets  $A$  and  $B$  is expressed by:

$$G(A, B) = \frac{|m(A \cup B)|}{|T|} = \frac{|m(A) \cap m(B)|}{|T|} = P(A, B),$$

which may be viewed as the joint probability of a transaction  $t$  containing itemsets  $A$  and  $B$ . The *absolute support* of an itemset  $B$  provided by  $A$  is defined by:

$$AS(A, B) = \frac{|m(A \cup B)|}{|m(A)|} = \frac{P(A, B)}{P(A)} = P(B \mid A).$$

This quantity shows the degree to which  $A$  implies  $B$ ,  $B$  depends on  $A$  or  $B$  associates with  $A$ . It may be viewed as the conditional probability of transactions containing the itemset  $B$  given that they contain the itemset  $A$ . The range of this measure is  $0 \leq AS(A, B) \leq 1$ . The *change support* of an itemset  $B$  provided by  $A$  is defined by:

$$CS(A, B) = AS(A, B) - G(B) = P(B | A) - P(B).$$

One may consider  $G(B)$  to be the prior probability of  $B$  and  $AS(A, B)$  the posterior probability of  $B$  after knowing  $A$ . The difference of posterior and prior probabilities represents the change of our confidence regarding whether  $A$  actually relates to  $B$ . The range of change support is from  $-1$  to  $1$ . For a positive value, one may say that  $A$  is positively related to  $B$ ; for a negative value, one may say that  $A$  is negatively related to  $B$ .

Three measures, absolute support  $AS(A, B)$ , generality  $G(A, B)$  and change support  $CS(A, B)$ , are used to define and quantify peculiarity rules and association rules [1,20]. We can qualitatively characterize association and peculiarity rules in the following table:

Rules	$AS(A, B)$	$G(A, B)$	$CS(A, B)$
Peculiarity Rules	High	Low	High
Association Rules	High	High	Low

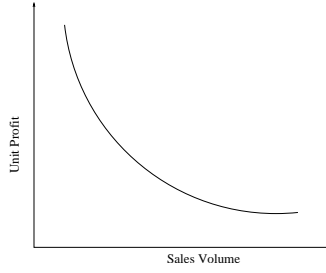
Generality  $G(A, B)$  and absolute support  $AS(A, B)$  are also called the support and confidence in association rule mining [1].

## 4 A Profit-Based Business Model

In this section, we propose a profit-based business model for evaluating mined rules. Profit is one of the primary financial objectives of a business [6]. It is one of basic tasks of business management. A general model of profit can be defined by: [6]

$$PF(A) = [f(A) - f'(A) - f''(A) - f'''(A)] \cdot V(A) - FOE, \quad (1)$$

where  $PF(A)$  is the total amount of profit on the itemset  $A$ ,  $f(A)$  is the sales price of the itemset  $A$ ,  $f'(A)$  represents the product costs of producing or purchasing the itemset  $A$ ,  $f''(A)$  represents unit-driven costs on the itemset  $A$ , such as the costs of shipping, handling and packaging,  $f'''(A)$  represents revenue-driven costs on the itemset  $A$ , such as commissions paid to salespersons and credit card discounts paid by retailers to the banks. These different costs are incremental (i.e. increased with each additional unit sold),  $V(A)$  is sales volume of  $A$  (i.e. total number of units of  $A$  actually are sold over a period), and  $FOE$  is fixed operating costs, such as rent, depreciation, salaries, and utilities. The amount of profit per unit after deducting various costs, including product costs, unit-driven costs and revenue-driven costs, from sales price is called unit margin or unit profit [6]. When the generality (probability) of selling itemset  $A$ ,  $G(A)$ , and total number of transactions  $|T|$  are known, the sales volume can be considered as



**Fig. 1.** Distribution between sales price and sales volume

$V(A) = |m(A)| = |T| \cdot G(A)$ . Let  $p(A)$  denote the unit profit, that is,  $p(A) = f(A) - f'(A) + f''(A) + f'''(A)$ . Then, the Equation 1 can be re-expressed simply as

$$PF(A) = p(A) \cdot V(A) - FOE. \quad (2)$$

In this paper, the fixed operating costs  $FOE$  are assumed to be unaffected by a marketing strategy. The other two factors, unit profit  $p(A)$  and sales volume  $V(A)$ , affect each other [9]. That is, when unit profit is increased, the sales volume is probably decreased. Conversely, when sales volume is increased, the unit profit probably has to be decreased. Their relationship can be demonstrated in Figure 1 [9]. Accordingly, marketing strategies to increase the profit can be generally classified into two groups based on the two factors in the profit model:

- **Price-based strategy:** to increase the unit profit (e.g. raising sales price, or reducing various costs).
- **Volume-based strategy:** to increase the sales volume (e.g. reducing sales price, or recommend and advertising items).

A measure, called *profit change*, can be defined to quantify the difference of the profits generated after and before applying a particular marketing strategy. Let  $s$  denote a particular marketing strategy. The profit change on itemset  $A$ , denoted as  $CP_s(A)$ , with respect to the marketing strategy  $s$  is formally defined by

$$\begin{aligned} CP_s(A) &= PF_s(A) - PF(A), \\ &= p_s(A) \cdot V_s(A) - p(A) \cdot V(A), \end{aligned} \quad (3)$$

where  $PF_s(A)$ ,  $p_s(A)$  and  $V_s(A)$  represent the profit, unit profit and sales volume on  $A$  after applying the marketing strategy  $s$ , respectively, and  $PF(A)$ ,  $p(A)$  and  $V(A)$  represent the profit, unit profit and sales volume on  $A$  before applying the marketing strategy, respectively. Obviously, the profit change on itemsets  $A$  and  $B$  can be expressed as:

$$\begin{aligned} CP_s(A, B) &= [PF_s(A) - PF(A)] + [PF_s(B) - PF(B)], \\ &= CP_s(A) + CP_s(B). \end{aligned} \quad (4)$$

The cost of running marketing strategy can be counted as various costs in the profit model. A positive profit change,  $CP_s(A, B) > 0$ , means the profit increased, and a

negative profit change,  $CP_s(A, B) < 0$ , states the profit decreased after applying the marketing strategy  $s$ .

## 5 An Analysis on Marketing Strategies, Discovered Rules and Profit Changes

Discovered rules can be used to develop the two types of marketing strategies for profit increasing. In this section, based on the profit-based business model, we investigate relationships between two types of rules and profit changes with respect to the price-based and volume-based marketing strategies.

### 5.1 Profit Changes Based on Price Changes

A price-based strategy is to raise the unit profits of items to increase the total profits, such as increasing sales price, reducing the unit costs, or cutting some services. With respect to a rule  $A \Rightarrow B$  and  $A \neq B$ , two types of price-based strategies are generally considered and analyzed as follows.

**Price-based strategy on itemset  $A$ :** Let  $s_1$  denote a price-based strategy on itemset  $A$ . This strategy probably leads to decrease the sales volume on the itemset  $A$ . The decreased sales volume and increased unit profit can be viewed as a kind of trade-off. Thus, we suppose that the profit on the itemset  $A$  after applying the strategy  $s_1$  is (at least) the same as the profit before applying the strategy, that is,  $CP_{s_1}(A) = 0$ . Then, the profit change  $CP_{s_1}(A, B)$  on itemsets  $A$  and  $B$  is

$$CP_{s_1}(A, B) = CP_{s_1}(B) = p(B) \cdot [V_{s_1}(B) - V(B)].$$

Suppose the association relationship between itemsets  $A$  and  $B$  is not changed. In other words, the absolute support  $AS(A, B)$  is not changed. Then, the difference of sales volumes on the itemset  $B$  can be expressed as

$$\begin{aligned} V_{s_1}(B) - V(B) &= V_{s_1}(A) \cdot AS(A, B) - V(A) \cdot AS(A, B), \\ &= [V_{s_1}(A) - V(A)] \cdot AS(A, B). \end{aligned}$$

Let  $\Delta V_{s_1}(B)$  denote the change of sales volume on itemset  $B$ , that is,  $\Delta V_{s_1}(B) = V_{s_1}(B) - V(B)$ , and  $\Delta V_{s_1}(A)$  denote the change of sales volume on itemset  $A$ , that is,  $\Delta V_{s_1}(A) = V_{s_1}(A) - V(A)$ . Then, the above equation can be expressed as

$$\Delta V_{s_1}(B) = \Delta V_{s_1}(A) \cdot AS(A, B). \quad (5)$$

and the profit change  $CP_{s_1}(A, B)$  is

$$CP_{s_1}(A, B) = p(B) \cdot \Delta V_{s_1}(A) \cdot AS(A, B). \quad (6)$$

Since the unit profit  $p(B)$  is a constant, the profit change  $CP_{s_1}(A, B)$  is determined by two factors: sales volume change  $\Delta V_{s_1}(A)$  and absolute support  $AS(A, B)$ . Therefore, sales volume change on itemset  $A$  determines the direction of profit change, while the



absolute support  $AS(A, B)$  determines the degree of profit change. A high absolute support can make profit change significantly.

Moreover, the sales volume  $V(B)$  can be expressed as the total number of transactions multiplying the generality of itemset  $B$ . Thus, the difference of sales volumes on itemset  $B$  can also be expressed as

$$\Delta V_{s_1}(B) = |T_{s_1}| \cdot G_{s_1}(B) - |T| \cdot G(B).$$

A large generality of itemset  $B$ ,  $G(B)$ , means a small space to decrease, while a low generality of itemset  $B$  means a large space to decrease.

We consider two types of rules, peculiarity rules [20] and association rules [1], with high absolute supports used in this type of price-based strategies. The relationships between profit change and the two types of rules are generally summarized in Table 1. From Table 1, we can obtain a general idea such that the profit decreasing based on the

**Table 1.** Relationships between profit changes and rules based on price increasing on  $A$

Discovered Rules	Probabilistic Interpretations					Profit Change
	$AS(A, B)$	$G(A, B)$	$CS(A, B)$	$G(A)$	$G(B)$	
Peculiarity Rules	High	Low	High	Low	Low	Low
Association Rules	High	High	Low	High	High	Very Low

association rules is more significant than the profit change based on the peculiarity rules when applying a price-based strategy on the itemset  $A$ .

*Example 1.* Suppose a supermarket uses a peculiarity rule,  $printer \Rightarrow ink$ , to develop a price-based strategy. The parameters of the rule are  $G(printer) = 2\%$ ,  $G(ink) = 1.6\%$ , and  $G(printer, ink) = 1.6\%$ ,  $AS(printer, ink) = 80\%$ ,  $CS(printer, ink) = 78.4\%$ . The unit profit and sales volume of *printer* are originally \$50.00 and 200. The unit profit and sales volume of *ink* are originally \$1.00 and 160. After applying the price-based strategy on *printer*, the unit profit of *printer* is raised to \$60.00, however, the sales volume of *printer* is down to 150. Suppose the sales of *ink* depends on the sales of *printer*, so the sales volume of *ink* will go down to 120 as the sales volume of *printer* going down. Therefore, the total profit of *printer* and *ink* is changed from \$10160 to \$9120, decreased about 10%.

Suppose a supermarket uses an association rule,  $milk \Rightarrow eggs$ , to develop a price-based strategy. The parameters of the rule are  $G(milk) = 80\%$ ,  $G(eggs) = 85\%$ , and  $G(milk, eggs) = 75\%$ ,  $AS(milk, eggs) = 94\%$ ,  $CS(milk, eggs) = 9\%$ . The unit profit and sales volume of *milk* are originally \$1.00 and 8,000. The unit profit and sales volume of *eggs* are originally \$0.50 and 8,500. After applying the price-based strategy on *milk*, the unit profit of *milk* is raised to \$1.20, however, the sales volume of *milk* is down to 5,000. Suppose the sales of *eggs* depends on the sales of *milk*, so the sales volume of *eggs* will go down to 5,000 as the sales volume of *milk* going down. Therefore, the total profit of *milk* and *eggs* is changed from \$12250 to \$8500, decreased about 31%.

**Price-based strategy on itemset  $B$ :** Let  $s_2$  denote a price-based strategy on itemset  $B$ . In this type of strategies, the unit profit and sales volume on the itemset  $A$  are supposed not to be changed. Thus, the profit change  $CP_{s_2}(A, B)$  is

$$CP_{s_2}(A, B) = CP_{s_2}(B) = p_{s_2}(B) \cdot V_{s_2}(B) - p(B) \cdot V(B).$$

If the rules show the larger dependence and larger generality of  $A$  and  $B$ , namely  $AS(A, B)$  and  $G(A, B)$  are very high, then increased unit profit on  $B$  may not make the sales volume decrease, otherwise, the unit profit increasing may lead to decrease the sales volume on itemset  $B$ .

The relationships between profit change and two types of rules are generally summarized in Table 2. It shows the association rules can be useful to generate more profit than the peculiarity rules.

**Table 2.** Relationships between profit change and rules based on price increasing on  $B$

Discovered Rules	Probabilistic Interpretations					Profit Change $CP_{s_2}(A, B)$
	$AS(A, B)$	$G(A, B)$	$CS(A, B)$	$G(A)$	$G(B)$	
Peculiarity Rules	High	Low	High	Low	Low	Low
Association Rules	High	High	Low	High	High	High

*Example 2.* Suppose a supermarket uses a peculiarity rule,  $printer \Rightarrow ink$ , to develop a price-based strategy. The parameters of the rule are  $G(printer) = 2\%$ ,  $G(ink) = 1.6\%$ , and  $G(printer, ink) = 1.6\%$ ,  $AS(printer, ink) = 80\%$ ,  $CS(printer, ink) = 78.4\%$ . The unit profit and sales volume of *printer* are originally \$50.00 and 200. The unit profit and sales volume of *ink* are originally \$1.00 and 160. After applying the price-based strategy on *ink*, the unit profit of *ink* is raised to \$1.50. Suppose the sales of *ink* depends on the sales of *printer*, so the sales volume of *ink* will keep unchanged. Therefore, the total profit of *printer* and *ink* is changed from \$10160 to \$10240, increased about 0.7%.

Suppose a supermarket uses an association rule,  $milk \Rightarrow eggs$ , to develop a price-based strategy. The parameters of the rule are  $G(milk) = 80\%$ ,  $G(eggs) = 85\%$ , and  $G(milk, eggs) = 75\%$ ,  $AS(milk, eggs) = 94\%$ ,  $CS(milk, eggs) = 9\%$ . The unit profit and sales volume of *milk* are originally \$1.00 and 8,000. The unit profit and sales volume of *eggs* are originally \$0.50 and 8,500. After applying the price-based strategy on *eggs*, the unit profit of *eggs* is raised to \$1.00. Suppose the sales of *eggs* depends on the sales of *milk*, so the sales volume of *eggs* will keep unchanged. Therefore, the total profit of *milk* and *eggs* is changed from \$12250 to \$16500, increased about 35%.

## 5.2 Profit Changes Based on Volume Changes

The volume-based strategies are to increase sales volumes (i.e. the amount of customers who buying items) to increase total profits. For example, one can reduce the sales price or recommend items to customers to increase sales volumes. With respect to a rule,  $A \Rightarrow B$  and  $A \neq B$ , two types of volume-based strategies are generally considered: strategy on the itemset  $A$  and strategy on the itemset  $B$ .

**Volume-based strategies on itemset  $A$ :** Let  $s_3$  denote a volume-based strategy on itemset  $A$ . The unit profit on itemset  $A$  probably has to be reduced in order to increase the sales volume. The increased sales volume and decreased unit profit are viewed as a kind of tradeoff. Suppose the profit on itemset  $A$  after applying a volume-based strategy is the same as the profit before applying the volume-based strategy. That is,  $PF_s(A) = PF(A)$  and  $CP_s(A) = 0$ . Then, the profit change  $CP_{s_3}(A, B)$  is

$$CP_{s_3}(A, B) = CP_{s_3}(B) = p(B) \cdot [V_{s_3}(B) - V(B)],$$

Suppose the association relationship between itemsets  $A$  and  $B$  is not changed, that is, the absolute support  $AS(A, B)$  is not changed. Then, the difference of sales volumes on the itemset  $B$  can be expressed as

$$\begin{aligned} V_{s_3}(B) - V(B) &= V_{s_3}(A) \cdot AS(A, B) - V(A) \cdot AS(A, B), \\ &= [V_{s_3}(A) - V(A)] \cdot AS(A, B). \end{aligned}$$

Let  $\Delta V_{s_3}(B)$  denote the change of sales volume on itemset  $B$ , that is,  $\Delta V_{s_3}(B) = V_{s_3}(B) - V(B)$ , and  $\Delta V_{s_3}(A)$  denote the change of sales volume on itemset  $A$ , that is,  $\Delta V_{s_3}(A) = V_{s_3}(A) - V(A)$ . Then, the above equation can be expressed as

$$\Delta V_{s_3}(B) = \Delta V_{s_3}(A) \cdot AS(A, B), \quad (7)$$

and the profit change  $CP_{s_3}(A, B)$  is

$$CP_{s_3}(A, B) = p(B) \cdot \Delta V_{s_3}(A) \cdot AS(A, B). \quad (8)$$

Since the unit profit  $p(B)$  is a constant, the profit change  $CP_{s_3}(A, B)$  is determined by two factors: sales volume change  $\Delta V_{s_3}(A)$  and absolute support  $AS(A, B)$ . Therefore, the sales volume on  $A$  determines the direction of the profit change, and the absolute support  $AS(A, B)$  determines the degree of profit decreasing. A high absolute support can make profit change significantly.

Moreover, the sales volume  $V(B)$  can be expressed as the total number of transactions multiplying the probability of selling the itemset  $B$ . Thus, the difference of sales volumes on the itemset  $B$  can also be expressed as

$$\Delta V_{s_1}(B) = |T_{s_1}| \cdot G_{s_1}(B) - |T| \cdot G(B).$$

A large generality of  $B$ ,  $G(B)$ , means a small space to decrease, while a low generality of  $B$  means a large space to decrease.

We consider two types of rules with high absolute supports used in this type of volume-based strategies. The relationships between profit change and the two types of rules are generally summarized in Table 3. Table 3 shows that the peculiarity rules can increase the profit more significantly than the association rules.

*Example 3.* Suppose a supermarket uses a peculiarity rule,  $printer \Rightarrow ink$ , to develop a volume-based strategy. The parameters of  $G(printer) = 2\%$ ,  $G(ink) = 1.6\%$ , and  $G(printer, ink) = 1.6\%$ ,  $AS(printer, ink) = 80\%$ ,  $CS(printer, ink) = 78.4\%$ . The unit profit and sales volume of *printer* are originally \$50.00 and 200. The unit

**Table 3.** Relationships between profit changes and Rules based on volume increasing on  $A$ 

Discovered Rules	Probabilistic Interpretations					Profit Change $CP_{s_3}(A, B)$
	$AS(A, B)$	$G(A, B)$	$CS(A, B)$	$G(A)$	$G(B)$	
Peculiarity Rules	High	Low	High	Low	Low	High
Association Rules	High	High	Low	High	High	Low

profit and sales volume of *ink* are originally \$1.00 and 160. After applying the volume-based strategy on *printer*, the unit profit of *printer* is reduced to \$5.00, and the sales volume of *printer* is up to 2000. Suppose the sales of *ink* depends on the sales of *printer*, so the sales volume of *ink* will go up to 1600 as the sales volume of *printer* going up. Therefore, the total profit of *printer* and *ink* is changed from \$10160 to \$11600, increased about 14%.

Suppose a supermarket uses an association rule,  $milk \Rightarrow eggs$ , to develop a volume-based strategy. The parameters of the rule are  $G(milk) = 80\%$ ,  $G(eggs) = 85\%$ , and  $G(milk, eggs) = 75\%$ ,  $AS(milk, eggs) = 94\%$ ,  $CS(milk, eggs) = 9\%$ . The unit profit and sales volume of *milk* are originally \$1.00 and 8,000. The unit profit and sales volume of *eggs* are originally \$0.50 and 8,500. After applying the volume-based strategy on *milk*, the unit profit of *milk* is reduced to \$0.90, and the sales volume of *milk* is up to 8,900. Suppose the sales of *eggs* depends on the sales of *milk*, so the sales volume of *eggs* will go up to 9,000. Therefore, the total profit of *milk* and *eggs* is changed from \$12250 to \$12510, increased about 2%.

**Volume strategy on itemset  $B$ :** Let  $s_4$  denote a volume-based strategy on itemset  $B$ . The unit profit and sales volume on the itemset  $A$  are supposed not to be changed. Thus, the profit change  $CP_{s_4}(A, B)$  is

$$CP_{s_4}(A, B) = CP_{s_4}(B) = p_{s_4}(B) \cdot V_{s_4}(B) - p(B) \cdot V(B).$$

If the discovered rules show the larger dependence and larger generality of  $A$  and  $B$ , namely  $AS(A, B)$  and  $G(A, B)$  are very high, then the decreased unit profit on itemset  $B$  may not make the sales volume increase significantly, otherwise, the unit profit decreasing may lead to increase the sales volume significantly. That means stronger the association is between two itemsets, the more profit is decreased when applying a volume-based strategy on itemset  $B$ .

Therefore, with a volume-based strategy, the relationships between profit changes and two types of rules are summarized in Table 4. Table 4 shows that by applying a

**Table 4.** Relationships between profit changes and Rules based on volume increasing on  $B$ 

Discovered Rules	Probabilistic Interpretations					Profit Change $CP_{s_4}(A, B)$
	$AS(A, B)$	$G(A, B)$	$CS(A, B)$	$G(A)$	$G(B)$	
Peculiarity Rules	High	Low	High	Low	Low	Low
Association Rules	High	High	Low	High	High	Very Low

volume-based strategy on itemset  $B$ , the association rules decrease the profit more significantly than the peculiarity rules.

*Example 4.* Suppose a supermarket uses a peculiarity rule,  $printer \Rightarrow ink$ , to develop a volume-based strategy. The parameters of the rule are  $G(printer) = 2\%$ ,  $G(ink) = 1.6\%$ , and  $G(printer, ink) = 1.6\%$ ,  $AS(printer, ink) = 80\%$ ,  $CS(printer, ink) = 78.4\%$ . The unit profit and sales volume of *printer* are originally \$50.00 and 200. The unit profit and sales volume of *ink* are originally \$1.00 and 160. After applying the volume-based strategy on *ink*, the unit profit of *ink* is down to \$0.90. Suppose the sales of *ink* depends on the sales of *printer*, so the sales volume of *ink* will keep unchanged. Therefore, the total profit of *printer* and *ink* is changed from \$10160 to \$10144, decreased about 0.2%.

Suppose a supermarket uses an association rule,  $milk \Rightarrow eggs$ , to develop a volume-based strategy. The parameters of the rule are  $G(milk) = 80\%$ ,  $G(eggs) = 85\%$ , and  $G(milk, eggs) = 75\%$ ,  $AS(milk, eggs) = 94\%$ ,  $CS(milk, eggs) = 9\%$ . The unit profit and sales volume of *milk* are originally \$1.00 and 8,000. The unit profit and sales volume of *eggs* are originally \$0.50 and 8,500. After applying the volume-based strategy on *eggs*, the unit profit of *eggs* is down to \$0.40. Suppose the sales of *eggs* depends on the sales of *milk*, so the sales volume of *eggs* will keep unchanged. Therefore, the total profit of *milk* and *eggs* is changed from \$12250 to \$11400, decreased about 7%.

Generally, based on the analysis above, we can summarize that different types of rules are useful for different marketing strategies. Based on the profit-based model, the association rules are more useful to develop price-based strategies, while the peculiarity rules are more useful to develop volume-based strategies.

## 6 Conclusion

The common practice in data mining focuses more on algorithmic and statistical aspects of rules. Although many useful results have been achieved, the real application of discovered rules remains to be a challenging problem. Without a domain specific semantic interpretation of rules, one may not solve this problem successfully. In this paper, we introduce a profit-based business model for the semantics study and application. Two types of rules and two types of marketing strategies are considered. We analyze the marketing strategies where the rules are used to generate high profits. One can find out that requirements of rules to generate high profits in different marketing strategies are different. It confirms that semantics and domain knowledge should be incorporated into data mining process.

## References

1. Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases, *Proceedings of ACM SIGMOD*, 207-216, 1993.
2. Braha, D. A theory of actionable data mining with application to semiconductor manufacturing control, *International Journal of Production Research*, to appear.

3. Coulehan, J.L. and Block, M.R. *The Medical Interview: Mastering Skills for Clinical Practice*, 5th Edition, F.A. Davis Co., Philadelphia, 2006.
4. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (Eds.), *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, California, 1996.
5. Fishburn, P.C. *Utility Theory for Decision Making*, John Wiley & Sons, Inc., New York, 1970.
6. Gardner, F.V. *Profit Management and Control*, McGraw-Hill Book Company, Inc., New York, 1955.
7. Geng, L. and Hamilton, H. Interestingness measures for data mining: A survey, *ACM Computing Surveys*, **38**, Article No. 9, 2006.
8. Han, J. and Kamber, M. *Data mining: Concept and Techniques*, Morgan Kaufmann, Palo Alto, CA, 2000.
9. Jobber, D. *Principles and Practice of Marketing*, Fourth edition, McGraw-Hill Europe, London, 2003.
10. Kleinberg, J., Papadimitriou, C. H. and Raghavan, P. A microeconomic view of data mining, *Data Mining and Knowledge Discovery*, **2**, 311C324, 1998.
11. Lin, T.Y., Yao, Y.Y. and Louie, E. Mining value added association rules, *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'02)*, 328-333, 2002.
12. Ling, C., Chen, T., Yang, Q. and Chen, J. Mining optimal actions for profitable CRM, *Proceedings of ICDM*, 767-770, 2002.
13. Liu, B., Hsu, W. and Ma, Y. Identifying non-actionable association rules, *Proceedings of KDD*, 329-334, 2001.
14. Ras, Z. and Gupta, S. Global action rules in distributed knowledge systems, in: L. Czaja, H.D. Burkhard, and P. Starke (Eds.), *Journal of Fundamenta Informaticae*, **51**, 175-184, 2002.
15. Roberts, F. *Measurement Theory*, Addison Wesley, Massachusetts, 1979.
16. Wang, K., Zhou, S. and Han, J. Profit mining: from patterns to actions, *Proceedings of the 8th International Conference on Extending Database Technology (EDBT'02)*, 70-87, 2002.
17. Weiss, G., Saar-Tsechansky, M. and Zadrozny, B. Report on UBDM-05: workshop on utility-based data mining. *SIGKDD Explorations*, **7**, 145-147, 2005.
18. Yao, Y.Y., Chen, Y.H. and Yang, X.D. A Measurement-theoretic foundation for rule interestingness evaluation, In: *Foundations and Novel Approaches in Data Mining Series: Studies in Computational Intelligence*, **9**, Lin, T.Y.; Ohsuga, S.; Liao, C.-J.; Hu, X. (Eds.), 41-59, 2005.
19. Yao, Y.Y., Zhao, Y. and Maguire, R.B. Explanation-oriented association mining using a combination of unsupervised and supervised learning algorithms, *Proceedings of the 16th Conference of the Canadian Society for Computational Studies of Intelligence (AI'03)*, 527-532, 2003.
20. Zhong, N. Yao, Y.Y. and Ohshima, M. Peculiarity oriented multidatabase mining, *IEEE Transactions on Knowledge and Data Engineering*, **15**, 952-960, 2003.

# Reasoning About Operations on Sets

Bernhard Heinemann

Fakultät für Mathematik und Informatik,  
FernUniversität in Hagen,  
58084 Hagen, Germany  
`bernhard.heinemann@fernuni-hagen.de`

**Abstract.** The present paper is about a framework of formal topological reasoning originating from the fundamental work of Moss and Parikh relating to this. We add a set of unary operators involving sets to that system. This new means of expression gives us considerably more expressive power with regard to spatial operations, but the accompanying logic remains sound and semantically complete with respect to the class of all subset spaces that are enriched accordingly. Moreover, the new logic turns out to be decidable. We prove these results by relying heavily on a particular extension of the common modal formalism, viz hybrid logic.

**Keywords:** the logic of knowledge, spatial logics, topological reasoning, set-valued functions, hybrid logic.

## 1 Introduction

The subject of this paper is the interplay between procedural knowledge and spatial concepts, in particular, those arising from topology. We extend some of the previous systems dealing with that, eg, those from [1], [2], and [3]. Our aim is to further clarify how the ideas of knowledge and space are related, and simultaneously develop a theoretical basis for corresponding reasoning tools. This is done in a way being similar to the one taken in [4], where the part state-valued functions play in that was investigated.

We begin the motivational part of the paper by recalling Moss and Parikh's particular approach in reasoning about knowledge; see [1,5]. The bi-modal *logic of knowledge and effort* invented in these papers facilitates on the one hand a qualitative description of procedures gaining knowledge, and reveals on the other hand a certain topological component of knowledge. In fact, since knowledge is represented by the space of all *knowledge states*, which are the sets of states an agent in question considers possible at a time, knowledge acquisition appears as a *shrinking procedure* regarding this space of sets. Thus certain notions from topology like *closeness* or *neighbourhood* turn up together with knowledge in a natural way.

Moss and Parikh suggestively called their system **topologic**. We prefer the acronym *MLS* here, indicating the modal language (and logic, respectively) of subset spaces. In the following, the basics of this language are given. As it has



just been indicated, formulas may contain two one-place operators: a modality  $K$  representing the knowledge of the agent under discussion and another one,  $\Box$ , representing (computational) effort. The domains for evaluating formulas are triples  $(X, \mathcal{O}, V)$  called *subset spaces*, which consist of a non-empty set  $X$  of states, a set  $\mathcal{O}$  of subsets of  $X$  representing the knowledge states of the agent, and a valuation  $V$  determining the states where the atomic propositions are true. The operator  $K$  then quantifies over some knowledge state  $U \in \mathcal{O}$ , whereas  $\Box$  quantifies ‘downward’ over  $\mathcal{O}$  since shrinking and acquiring knowledge correspond to each other.

After [1] was published, several classes of subset spaces, including the ordinary topological ones, could be characterized by means of *MLS*; cf [6,7,8,9]. However, more expressive power is needed to specify other interesting topological notions like separation or connectedness. To this end, a version of *MLS* based on hybrid logic, *hybrid MLS*, was introduced in [2] and further developed subsequently; see [3]. The origins of hybrid logic date back from the work of A. Prior on temporal logic. Corresponding features have been successfully applied to many fields since then, in particular, to various formalisms concerning space and time; see [10] for an overview. Actually, *hybrid MLS* turned out to be quite useful as well. This is demonstrated, eg, in the paper [11], where a problem raised in [9] could be solved by using hybrid methods. In the present paper, we utilize that logic for proving our main results.

Quite recently, state-valued functions have been incorporated into *hybrid MLS*. In the paper relating to this, [4], the main concern was a formal description of the relationship between knowledge and *continuity*. Here we focus on a complementary aspect of (*hybrid*) *MLS* by considering unary operations on *sets*. Forming the complement, the closure, or the interior, of a set are examples of such operations. Thus further-reaching topological modelling and reasoning is supported by the extended system.

The subsequent technical part of the paper is organized as follows. In the next section we define the language for *MLS* with operations on sets. We give also some examples concerning expressiveness there. In Section 3, we turn to the arising logic. Hybridizing the underlying language enables us to establish a corresponding soundness and completeness theorem. In Section 4, we prove that the new logic is decidable. Concluding the paper, we summarize and mention future research.

The theory developed below fits into the chapter on topology and epistemic logic of the forthcoming handbook [12], in which several alternative approaches to reasoning about space are treated as well.

## 2 The Language

We now introduce the language underlying the logics studied later on. After that, we use the new language to describe certain properties of some spatial and topological operations, respectively.

We first define the syntax. Let  $\text{PROP} = \{A, B, \dots\}$  be a denumerable set of symbols called *proposition letters*. Moreover, let  $\mathcal{F} = \{F, G, \dots\}$  be a set of



one-place function symbols (mostly called *operation symbols* below). Then, the set WFF of *well-formed formulas* over  $\text{PROP} \cup \mathcal{F}$  is defined by the rule

$$\alpha ::= A \mid \neg\alpha \mid \alpha \wedge \beta \mid K\alpha \mid \Box\alpha \mid [F]\alpha.$$

The missing boolean connectives are treated as abbreviations, as needed. The duals of the modal operators  $K$ ,  $\Box$  and  $[F]$  are denoted  $L$ ,  $\Diamond$  and  $\langle F \rangle$ , respectively.<sup>1</sup>

Second, we turn to semantics. For a start, we define the relevant domains. We let  $\mathcal{P}(X)$  designate the powerset of a given set  $X$ .

**Definition 1 (Subset frames and subset spaces)**

1. A subset frame is a pair  $\mathcal{S} := (X, \mathcal{O})$ , where  $X$  is a non-empty set and  $\mathcal{O} \subseteq \mathcal{P}(X)$  a set of subsets of  $X$  such that  $\{\emptyset, X\} \subseteq \mathcal{O}$ .
2. Let  $\mathcal{S} = (X, \mathcal{O})$  be a subset frame. The set of neighbourhood situations of  $\mathcal{S}$  is  $\mathcal{N}_{\mathcal{S}} := \{(x, U) \mid x \in U \text{ and } U \in \mathcal{O}\}$ .
3. Let  $\mathcal{S} = (X, \mathcal{O})$  be a subset frame and  $V : \text{PROP} \longrightarrow \mathcal{P}(X)$  a mapping. Then  $V$  is called an  $\mathcal{S}$ -valuation.
4. Let  $\mathcal{S} = (X, \mathcal{O})$  be a subset frame and  $V$  an  $\mathcal{S}$ -valuation. Then,  $\mathcal{M} := (X, \mathcal{O}, V)$  is called a subset space. In this case we say that  $\mathcal{M}$  is based on  $\mathcal{S}$ .

In essence, Definition 1 is like the original one from [5]. Because of the forward looking nature of the effort operator, the requirement ‘ $\{\emptyset, X\} \subseteq \mathcal{O}$ ’ is in a sense insignificant, but convenient for *MLS*; cf [5], Sec. 1.1. That is no longer true for the extended system; see Sec. 3. However, we must be careful with  $X$  and the empty set when dealing with operations on sets; cf Example 1 below.

**Definition 2 (Subset frames and subset spaces with operations)**

1. A subset frame with operations is a triple

$$\mathcal{S} := (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\}),$$

where  $(X, \mathcal{O})$  is a subset frame and, for every  $F \in \mathcal{F}$ ,  $f_F : \mathcal{O} \longrightarrow \mathcal{O}$  is a partial function.

2. Let  $\mathcal{S} = (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\})$  be a subset frame with operations and  $V$  an  $\mathcal{S}$ -valuation. Then,  $\mathcal{M} := (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\}, V)$  is called a subset space with operations (or, in short, an SSO).

The use of partial functions is adequate for interpreting the elements of  $\mathcal{F}$  since  $\mathcal{O}$  need not be closed under some of the operations that are to be modelled.

Now, let an SSO  $\mathcal{M}$  be given. We define the relation of satisfaction,  $\models_{\mathcal{M}}$ , between neighbourhood situations of the underlying frame and well-formed formulas. In the following, neighbourhood situations are written without brackets.

<sup>1</sup> The dual of, eg,  $K$  is given by  $L\alpha \equiv \neg K\neg\alpha$ , for all  $\alpha \in \text{WFF}$  (and correspondingly for the other operators). Semantically, the dual represents the ‘existential shape’ of the modality.

**Definition 3 (Satisfaction and validity).** *Let*

$$\mathcal{M} := (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\}, V)$$

*be an SSO based on  $\mathcal{S} = (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\})$ , and let  $x, U \in \mathcal{N}_{\mathcal{S}}$  be a neighbourhood situation. Then*

$$\begin{aligned} x, U \models_{\mathcal{M}} A & : \iff x \in V(A) \\ x, U \models_{\mathcal{M}} \neg\alpha & : \iff x, U \not\models_{\mathcal{M}} \alpha \\ x, U \models_{\mathcal{M}} \alpha \wedge \beta & : \iff x, U \models_{\mathcal{M}} \alpha \text{ and } x, U \models_{\mathcal{M}} \beta \\ x, U \models_{\mathcal{M}} K\alpha & : \iff \text{for all } y \in U : y, U \models_{\mathcal{M}} \alpha \\ x, U \models_{\mathcal{M}} \Box\alpha & : \iff \text{for all } U' \in \mathcal{O} : (x \in U' \subseteq U \Rightarrow x, U' \models_{\mathcal{M}} \alpha) \\ x, U \models_{\mathcal{M}} [F]\alpha & : \iff \text{for all } x', U' \in \mathcal{N}_{\mathcal{S}} : (f_F(U) = U' \Rightarrow x', U' \models_{\mathcal{M}} \alpha), \end{aligned}$$

*for all  $A \in \text{PROP}$ ,  $F \in \mathcal{F}$ , and  $\alpha, \beta \in \text{WFF}$ . In case  $x, U \models_{\mathcal{M}} \alpha$  is true we say that  $\alpha$  holds in  $\mathcal{M}$  at the neighbourhood situation  $x, U$ . Furthermore, a formula  $\alpha$  is called valid in  $\mathcal{M}$  iff it holds in  $\mathcal{M}$  at every neighbourhood situation. (Manner of writing:  $\mathcal{M} \models \alpha$ .)*

The following comments on this definition may be useful. We will refer to some of the points addressed therein later on.

1. Apart from the last clause the definition clearly applies to subset spaces in the sense of Definition 1.
2. The meaning of proposition letters is independent of neighbourhoods by definition, thus ‘stable’ with respect to  $\Box$ . This fact is reflected by a special axiom; see Sec. 3.
3. According to the above definitions we realized operations on sets *implicitly*, i.e., by means of a respective modality. The explicit way, i.e., by using first-order-like terms, turns out to be unsuitable for our purposes.<sup>2</sup>

Concluding this section we give some sample specifications. These concern the operations we already mentioned in the introduction.

*Example 1 (Complementation).* Let  $\mathcal{S} = (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\})$  be a subset frame with operations. Assume that  $\mathcal{O}$  is closed under complements, and let  $G \in \mathcal{F}$  be an operation symbol which is to model complementation. Then  $f_G$  is inverse to itself, i.e., satisfies  $f_G \circ f_G = \text{id}_{\mathcal{O}}$ . Thus  $\mathcal{M} \models K\alpha \rightarrow [G][G]\alpha$  is true for all SSOs  $\mathcal{M}$  based on  $\mathcal{S}$  and all formulas  $\alpha \in \text{WFF}$ . Moreover, we also have that  $\mathcal{M} \models \alpha \rightarrow [G]\langle G \rangle\alpha$  is true for all such  $\mathcal{M}$  and  $\alpha$ . That is, the accessibility relation associated with the modality  $[G]$  is *symmetric*. Note, however, that although  $f_G$  is a total function that relation is not *serial* (cf [14], § 1) since the schema  $[G]\alpha \rightarrow \langle G \rangle\alpha$  is not sound. The latter is due to the fact that  $\{\emptyset, X\} \subseteq \mathcal{O}$ ; see the remark right before Definition 2.

<sup>2</sup> An attempt to ‘algebraizing’ hybrid logic in this spirit was undertaken, eg, in [13].

*Example 2 (Interior and closure).* During this example we assume that the underlying subset frame is appropriately derived from a topological space. If  $f_G$  models the assignment of the interior (or the closure) to a subset, where  $G$  is any operation symbol, then obviously the following formulas are valid:  $[G]\alpha \rightarrow [G][G]\alpha$  and  $[G]([G]\alpha \rightarrow \alpha)$ , for all  $\alpha \in \text{WFF}$ . The first schema expresses that the accessibility relation associated with  $[G]$  is *transitive*, and the second one says that it is *secondary reflexive*; cf [15], 3.51. Additionally, we can describe how  $[G]$  interacts with the ‘composite modality’  $K\Box$  in these special cases. We actually have that the schemata  $K\Box\alpha \rightarrow [G]\alpha$  and  $\alpha \rightarrow [G]L\Diamond\alpha$  are valid for the interior and the closure, respectively. – The point of view taken up in this example is complementary to the usual one; cf, eg, [16] or the chapter on modal logics of space of the handbook [12].

A more important example follows at the end of the next section.

### 3 The Logic

In this section, we first propose an axiomatization of the logic of subset spaces with operations. The logic is designated *MLSO*. Secondly, we hybridize this logic and show the soundness and semantic completeness of the resulting hybrid version, *HLSO*. And finally, the usefulness of the hybrid language for applications is demonstrated by an example.

To begin with, we list the axioms for *MLS* from [5].

1. All instances of propositional tautologies.
2.  $K(\alpha \rightarrow \beta) \rightarrow (K\alpha \rightarrow K\beta)$
3.  $K\alpha \rightarrow (\alpha \wedge KK\alpha)$
4.  $L\alpha \rightarrow KL\alpha$
5.  $\Box(\alpha \rightarrow \beta) \rightarrow (\Box\alpha \rightarrow \Box\beta)$
6.  $(A \rightarrow \Box A) \wedge (\Diamond A \rightarrow A)$
7.  $\Box\alpha \rightarrow (\alpha \wedge \Box\Box\alpha)$
8.  $K\Box\alpha \rightarrow \Box K\alpha$ ,

where  $A \in \text{PROP}$  and  $\alpha, \beta \in \text{WFF}$ . In this way, it is expressed that for every Kripke model  $M$  validating these axioms

- the accessibility relation  $\xrightarrow{K}$  of  $M$  belonging to the knowledge operator is an equivalence,
- the accessibility relation  $\xrightarrow{\Box}$  of  $M$  belonging to the effort operator is reflexive and transitive,
- the composite relation  $\xrightarrow{\Box} \circ \xrightarrow{K}$  is contained in  $\xrightarrow{K} \circ \xrightarrow{\Box}$  (this is usually called the *cross property*), and
- the valuation of  $M$  is constant along every  $\xrightarrow{\Box}$ -path, for all propositional letters; see the second remark following Definition 3.

The cross property is induced by Axiom 8, which is called the *Cross Axiom* therefore. The Cross Axiom (or a variant of it) is characteristic of every logic including knowledge and an effort-like modality.

Now let  $\mathcal{F}$  be a fixed set of operation symbols. It will be convenient to assume that  $\mathcal{F}$  is finite. The new axioms describing operations on sets read as follows.

- (O 1)  $[F](\alpha \rightarrow \beta) \rightarrow ([F]\alpha \rightarrow [F]\beta)$
- (O 2)  $K[F](\alpha \rightarrow L\beta) \vee K[F](\beta \rightarrow L\alpha)$
- (O 3)  $[F]\alpha \rightarrow [F]K\alpha \wedge K[F]\alpha,$

where  $F \in \mathcal{F}$  and  $\alpha, \beta \in \text{WFF}$ . A couple of points are worth mentioning here. First, note that capturing the functionality of  $f_F$  works completely different from the case of state-valued functions; cf [4]. In particular, contrasting that case it is possible to give a purely modal description here (although the type of  $f_F$  is ‘higher’). And second, Axiom (O 2) is related to the modal formula schema corresponding to *no branching in the future* of the accessibility relation; cf [14], p 22. This axiom was used literally (but with respect to a different semantics) as a certain linearity constraint for ‘temporal’ *MLS*; cf [17].

We obtain the logical system *MLSO* by adding the standard proof rules of modal logic, i.e., *modus ponens* and *necessitation with respect to each modality*.

**Definition 4 (The logic).** *Let MLSO be the smallest set of formulas which contains the axiom schemata 1 – 8 and (O 1) – (O 3) and is closed under the application of the following rule schemata:*

$$(\text{MODUS PONENS}) \quad \frac{\alpha \rightarrow \beta, \alpha}{\beta} \quad (\Delta\text{-NECESSITATION}) \quad \frac{\alpha}{\Delta\alpha},$$

where  $\alpha, \beta \in \text{WFF}$  and  $\Delta \in \{K, \Box\} \cup \{[F] \mid F \in \mathcal{F}\}$ .

We could now use the established techniques for proving the completeness of *MLSO* with respect to the class of all SSOs; cf [5], Sec. 2.2. However, the same does not go any longer for decidability, due to the schema (O 2). Thus we extend the methods of proof right away in order to obtain both results.

To this end, we enrich the language with elements from hybrid logic. Let  $N_{\text{stat}} = \{i, j, \dots\}$  and  $N_{\text{sets}} = \{I, J, \dots\}$  be two disjoint denumerable sets of symbols called *names of states* and *names of sets*, respectively. The elements of  $N_{\text{stat}} \cup N_{\text{sets}}$  are also called *nominals*. Moreover, let **A** designate the *global modality* (cf [18], Sec. 7.1) and **E** its dual.<sup>3</sup>

**Definition 5 (Hybrid SSOs).** *Let  $\mathcal{S} = (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\})$  be a subset frame with operations.*

1. *A mapping  $V : \text{PROP} \cup N_{\text{stat}} \cup N_{\text{sets}} \longrightarrow \mathcal{P}(X)$  is called a hybrid  $\mathcal{S}$ -valuation iff the following two conditions are satisfied:*
  - (a)  *$V(i)$  is either  $\emptyset$  or a singleton subset of  $X$  for every  $i \in N_{\text{stat}}$ , and*

<sup>3</sup> We need **A** mainly for technical reasons; see the remark following Definition 6.

- (b)  $V(I) \in \mathcal{O}$  for every  $I \in N_{sets}$ .
2. Let  $V$  be a hybrid  $\mathcal{S}$ -valuation. Then,  $\mathcal{M} := (X, \mathcal{O}, V)$  is called a hybrid subset space with operations (or, in short, an HSSO).

Note that the definition takes into account that nominals may have an empty denotation. This is appropriate to us for technical reasons, but not usual for standard hybrid logic.

Now, we extend Definition 3 accordingly.

**Definition 6 (Hybrid satisfaction and validity).** *Let*

$$\mathcal{M} = (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\}, V)$$

*be an HSSO based on  $\mathcal{S} = (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\})$ , and let  $x, U \in \mathcal{N}_{\mathcal{S}}$  be a neighbourhood situation. Then*

$$\begin{aligned} x, U \models_{\mathcal{M}} i & : \iff x \in V(i) \\ x, U \models_{\mathcal{M}} I & : \iff V(I) = U \\ x, U \models_{\mathcal{M}} A\alpha & : \iff \text{for all } x', U' \in \mathcal{N}_{\mathcal{S}} : x', U' \models_{\mathcal{M}} \alpha, \end{aligned}$$

*for all  $i \in N_{stat}$ ,  $I \in N_{sets}$  and  $\alpha \in \text{WFF}$ .<sup>4</sup> In case  $x, U \models_{\mathcal{M}} \alpha$  is true we say that  $\alpha$  holds in  $\mathcal{M}$  at the neighbourhood situation  $x, U$ , as above. Additionally, a formula  $\alpha$  is called valid in  $\mathcal{M}$  iff it holds in  $\mathcal{M}$  at every neighbourhood situation.*

Note that the formulas of the form  $i \wedge I$ , where  $i \in N_{stat}$  and  $I \in N_{sets}$ , can be taken as names of neighbourhood situations. The hybrid *satisfaction operator* associated with such a name (cf [18], Sec. 7.3) then reads  $@_{i \wedge I} \alpha := E(i \wedge I \wedge \alpha)$ , where  $\alpha \in \text{WFF}$ . Thus a formula of that kind functions as a ‘proper’ nominal in SSOs. The operators  $@_{i \wedge I}$  are needed for axiomatizing the logic of HSSOs; cf footnote 3.

Actually, we do not want to list the axioms for hybrid *MLS* here, but refer the reader to the paper [3] regarding this. However, one crucial axiom of the new system cannot be too strongly emphasized, viz

$$(O2') \quad \langle F \rangle (i \wedge I) \wedge L \langle F \rangle (j \wedge J) \rightarrow [F] ((i \wedge I) \rightarrow L(j \wedge J)),$$

where  $i, j \in N_{stat}$  and  $I, J \in N_{sets}$ . This schema serves as a substitute for Axiom (O2) in the hybrid context and contributes decisively to the final success of our approach.

Furthermore, the unorthodox proof rules of the new system are worth mentioning.

**Definition 7 (Hybrid proof rules).** *The following rule schemata have to be added to the modal ones:*

$$(NAME_{stat}) \quad \frac{j \rightarrow \beta}{\beta} \quad (NAME_{sets}) \quad \frac{J \rightarrow \beta}{\beta}$$

<sup>4</sup> From now on, WFF denotes the set of formulas of the enriched language.

$$(\nabla\text{-ENRICHMENT}) \quad \frac{E(i \wedge I \wedge \nabla(j \wedge J \wedge \alpha)) \rightarrow \beta}{E(i \wedge I \wedge \nabla \alpha) \rightarrow \beta},$$

where  $\alpha, \beta \in \text{WFF}$ ,  $i, j \in \mathbf{N}_{stat}$ ,  $I, J \in \mathbf{N}_{sets}$ ,  $\nabla \in \{\mathbf{L}, \diamond, \mathbf{E}\} \cup \{\langle F \rangle \mid F \in \mathcal{F}\}$ , and  $j, J$  are new each time (i.e., do not occur in any other syntactic building block of the respective rule).

For the reader not familiar with hybrid proof rules a ‘contrapository’ reading is suggested; eg, the rule  $(\text{NAME}_{stat})$  is to be read ‘if  $\beta$  is satisfiable, then  $j \wedge \beta$  is satisfiable, too’ (provided that the nominal  $j$  does not occur in  $\beta$ ). From that, the soundness of the unorthodox rules should be rather apparent.

Technically, the NAME and ENRICHMENT rules have to be used for proving an appropriate *Lindenbaum Lemma* and an *Existence Lemma*, respectively; cf [18], Lemmata 7.25 and 7.27, and [3], Lemmata 3.3 and 3.6. Both auxiliary results make up the first steps towards the desired completeness theorem. In fact, these lemmata enable us to ‘hybridize’ the canonical model of the logical system. The model falsifying a given non-derivable formula then can be obtained as a certain space of partial functions,  $X$ , over the carrier set  $D$  of that model, actually in the following way. The domain  $\text{dom}(h)$  of every function  $h \in X$  is a maximal subset of the set  $\mathcal{Q} := \{[\Sigma] \mid \Sigma \in D\}$  of all equivalence classes  $[\Sigma] := \{\Sigma' \in D \mid \Sigma \xrightarrow{K} \Sigma'\}$  of the accessibility relation induced by the modality  $K$ . Maximality is meant with regard to the following two conditions here:

1.  $h([\Sigma]) \in [\Sigma]$  for all  $[\Sigma] \in \text{dom}(h)$ , and
2.  $h([\Sigma]) \xrightarrow{\square} h([\Theta])$  for all  $[\Sigma], [\Theta] \in \text{dom}(h)$  such that  $[\Sigma] \preceq [\Theta]$ ;

the *precedence relation*  $\preceq$  occurring in the second item is defined by  $[\Sigma] \preceq [\Theta] : \iff \exists \Sigma' \in [\Sigma], \Theta' \in [\Theta] : \Sigma' \xrightarrow{\square} \Theta'$ , where  $\Sigma, \Theta$  are points of  $D$  and  $\xrightarrow{\square}$  denotes (as above) the accessibility relation belonging to the effort modality  $\square$ . We write  $h_{\Sigma} := h([\Sigma])$  in case  $h([\Sigma])$  exists. Furthermore, we let

- $U_{[\Sigma]} := \{h \in X \mid h_{\Sigma} \text{ exists}\}$ , for all  $\Sigma \in D$ ,
- $\mathcal{O} := \{U_{[\Sigma]} \mid \Sigma \in D\} \cup \{\emptyset, X\}$ , and
- $V : \text{PROP} \cup \mathbf{N}_{stat} \cup \mathbf{N}_{sets} \longrightarrow \mathcal{P}(X)$  be defined by

$$h \in V(c) : \iff c \in h_{\Sigma} \text{ for some } \Sigma \in D \text{ such that } h_{\Sigma} \text{ exists,}$$

for all  $c \in \text{PROP} \cup \mathbf{N}_{stat} \cup \mathbf{N}_{sets}$ .

With that, we obtain the relevant *Truth Lemma*, from which completeness follows in case no operation symbols are involved.

**Lemma 1.**  $\mathcal{S} := (X, \mathcal{O})$  is a subset frame and  $V$  a hybrid  $\mathcal{S}$ -valuation. Moreover, letting  $\mathcal{M} := (X, \mathcal{O}, V)$  we have that  $(h, U_{[\Sigma]} \models_{\mathcal{M}} \alpha \iff \alpha \in h_{\Sigma})$  is valid for all formulas  $\alpha$  in which no operation symbols occur, every function  $h \in X$ , and all points  $\Sigma \in D$  such that  $h \in U_{[\Sigma]}$ .<sup>5</sup>

<sup>5</sup> As to this satisfaction relation, cf the first remark following Definition 3.

We now define the interpretation  $f_F : \mathcal{O} \longrightarrow \mathcal{O}$  of the operation symbol  $F$ , for every  $F \in \mathcal{F}$ . Let  $U_{[\Sigma]}$  be any element of  $\mathcal{O}$ , where  $\Sigma \in D$ . We distinguish the cases whether or not there exists some  $\Theta \in D$  such that  $\Sigma \xrightarrow{[F]} \Theta$ . According to that we let  $f_F(U_{[\Sigma]}) := U_{[\Theta]}$ , if  $\Theta$  exists, and be undefined otherwise. As the following lemma shows, every  $F \in \mathcal{F}$  is in fact realized as a certain functional of a higher type in this way.

**Lemma 2.**  *$f_F$  is well-defined.*

*Proof.* Some results from [3] are required for the proofs of this and the following lemma. These proofs are directed to the expert. – We have to show that the definition of  $f_F$  is independent of  $\Sigma$  and  $\Theta$ , respectively. We may restrict attention to the case  $f_F$  is defined. First, let  $\Sigma' \in D$  be such that  $U_{[\Sigma]} = U_{[\Sigma']}$ . Then  $[\Sigma] = [\Sigma']$  holds according to [3], Lemma 3.8.1 and Lemma 3.10. That is,  $\Sigma \xrightarrow{\kappa} \Sigma'$ . – Second, take any  $\Theta' \in D$  such that  $\Sigma' \xrightarrow{[F]} \Theta'$ . We then must show that  $\Theta \xrightarrow{\kappa} \Theta'$  is valid. For this we utilize that every element of  $D$  is *named*, i.e., determined by some formula  $i \wedge I$  where  $i \in N_{stat}$  and  $I \in N_{sets}$  (cf [3], Lemma 3.5.3). Let, correspondingly,  $i \wedge I \in \Theta$  and  $j \wedge J \in \Theta'$ . Then,  $\langle F \rangle(i \wedge I) \wedge \mathsf{L}\langle F \rangle(j \wedge J) \in \Sigma$ . Because of Axiom (O2') it follows that  $[F]((i \wedge I) \rightarrow \mathsf{L}(j \wedge J)) \in \Sigma$ , hence  $\mathsf{L}(j \wedge J) \in \Theta$ . Consequently,  $\Theta \xrightarrow{\kappa} \Theta'$ , as desired. We obtain that  $[\Theta] = [\Theta']$  and, therefore,  $U_{[\Theta]} = U_{[\Theta']}$ .

As a consequence of Lemma 2 we get that  $\mathcal{M} := (X, \mathcal{O}, \{f_F \mid F \in \mathcal{F}\}, V)$  is an HSSO. It remains to establish the *Truth Lemma* for the case involving operations on sets.

**Lemma 3.** *Lemma 1 remains valid if operations on sets are integrated into the system.*

*Proof.* In this proof we use, among other things, that a function  $h \in X$  is already determined by its value for a single argument. And, vice versa, every  $\Sigma \in D$  induces an element of  $X$  passing through this point; see [3], Lemma 3.9. Therefore, that function is denoted  $h^\Sigma$ . – Let  $\alpha = \langle F \rangle \beta$ . We first prove the right-to-left direction. Assume that  $\alpha \in h_\Sigma$  is valid. Then there exists some  $\Theta \in D$  such that  $h_\Sigma \xrightarrow{[F]} \Theta$  and  $\beta \in \Theta$ . Using the above notations we let  $g := h^\Theta$  and get  $\Theta = g_\Theta$  with that. It then follows that  $g \in U_{[\Theta]}$ . From the induction hypothesis we obtain  $g, U_{[\Theta]} \models_{\mathcal{M}} \beta$ . But  $U_{[\Theta]} = f_F(U_{[\Sigma]})$ , by definition. Thus we have that  $h, U_{[\Sigma]} \models_{\mathcal{M}} \langle F \rangle \beta$ , according to the last clause of Definition 3.

Conversely, let  $h, U_{[\Sigma]} \models_{\mathcal{M}} \langle F \rangle \beta$  be satisfied. By Definition 3, there are  $g \in X$  and  $\Theta \in D$  such that  $g \in U_{[\Theta]}$ ,  $U_{[\Theta]} = f_F(U_{[\Sigma]})$  and  $g, U_{[\Theta]} \models_{\mathcal{M}} \beta$ . By applying the induction hypothesis we obtain  $\beta \in g_\Theta$  from that. Now, it remains to prove that  $h_\Sigma \xrightarrow{[F]} g_\Theta$  is valid. Here is the place where Axiom (O3) comes into play. But first of all we conclude from the condition  $U_{[\Theta]} = f_F(U_{[\Sigma]})$  and the definition of  $f_F$  that there is some  $\Xi \in D$  such that  $\Sigma \xrightarrow{[F]} \Xi$  and  $U_{[\Xi]} = U_{[\Theta]}$ , i.e.,  $\Xi \xrightarrow{\kappa} \Theta$

(see the proof of Lemma 2). We also have that  $\Theta \xrightarrow{\mathbf{K}} g_\Theta$ . Thus the first part of Axiom (O3) implies that  $\Sigma \xrightarrow{[F]} g_\Theta$ . Since  $h_\Sigma \xrightarrow{\mathbf{K}} \Sigma$  is true we finally obtain  $h_\Sigma \xrightarrow{[F]} g_\Theta$  with the aid of the second part of Axiom (O3). This proves the left-to-right direction.

The following theorem can easily be obtained from the previous lemmata.

**Theorem 1 (Soundness and completeness).** *A formula  $\alpha \in \text{WFF}$  is valid in all SSOs, iff  $\alpha$  is contained in HLSO.*

It should be mentioned that the hybrid framework is not only useful for theoretical reasons but also for concrete applications. We give an example.

*Example 3 (Monotone functions).* The new language enables us to specify the *monotonicity* of functions operating on sets. Monotone set-valued functions are important in connection with the fixpoint characterization of the semantics of certain language constructs, according to the *Knaster-Tarski Fixpoint Theorem*. The formula schema capturing the monotonicity of  $f_F : \mathcal{O} \longrightarrow \mathcal{O}$  reads

$$\Diamond \langle F \rangle I \rightarrow \langle F \rangle \Diamond I,$$

where  $I \in \mathbf{N}_{sets}$ . Note the formal similarity of this schema to the dual of the Cross Axiom. Moreover, note that it is true the corresponding modal schema,  $\Diamond \langle F \rangle \alpha \rightarrow \langle F \rangle \Diamond \alpha$ , is sound for monotone functions, but it is not a correspondent in the spirit of modal logic; cf [14], § 1.

## 4 Decidability

We now argue that *HLSO* is decidable. This will imply that *MLSO* is decidable as well. At the end of this section we comment on the complexity of the *HLSO*-satisfiability problem.

**Theorem 2 (Decidability).** *HLSO is a decidable set of formulas.*

*Proof.* It is our aim to establish a certain finite model property of the logic. This can be done with the aid of *filtrations*; cf [14], § 4. In order to validate the axioms in the filtrated model, Axiom (O2') in particular, we first of all must fix the filter set suitably. The nominals as well as the operation symbols occurring in the (consistent) formula  $\alpha$  for which we want to find a finite model have to be taken into account for that. In concrete terms, the following sets of formulas have to be added to the set  $\Sigma_0$  we started off with in case of the function-free language (cf [3], Sec. 4):  $\Sigma_1 := \{[F] \neg \top \mid F \in \mathcal{F} \text{ occurs in } \alpha\}$ , and  $\Sigma_2 := \{[F](i \wedge I), [F] \neg (i \wedge I) \mid F \in \mathcal{F}, i \in \mathbf{N}_{stat} \text{ and } I \in \mathbf{N}_{sets} \text{ occur in } \alpha\}$ . We then close the resulting set of formulas successively under negations, finite conjunctions of pairwise distinct elements, single applications of the operator  $\mathbf{L}$ , and, finally, formation of subformulas. Thus the new filter set too is finite and subformula closed.



We secondly take the respective *smallest* filtration of all the accessibility relations of the canonical model (where  $\alpha$  is realized at some point); cf [14], p 33. We then modify the structure obtained in this way with regard to the nominals and the modalities  $[F]$ , respectively, which do *not* occur in  $\alpha$ . We take the nowhere defined function as the interpretation of an operation symbol not occurring in  $\alpha$ , and the empty set as the denotation of such a nominal. This modification obviously does not affect the semantics of  $\alpha$ .

The model surgery procedure which just has been indicated provides for the desired validity of Axiom (O2') (and of some other hybrid axioms not considered here). In fact, by exploiting both the minimality of the filtrations and the definition of the sets  $\Sigma_1$  and  $\Sigma_2$ , respectively, we can directly calculate in case all the names and operations involved in the axiom really occur in  $\alpha$ ; otherwise the proof is even simpler.<sup>6</sup>

Finally, we mention that the validation of Axiom (O3) is similar to the one of the Cross Axiom in [5], proof of Lemma 2.10. The closure properties of the filter set are utilized for that.

The finite model property of the logic *HLSO* with respect to a certain recursively enumerable set of models can be inferred from that in a standard manner now. (An initial segment of the natural numbers can be chosen as the carrier set of every model from that set, actually.) This gives us the desired decidability result, in accordance with [18], Theorem 6.13.

The decidability of *MLSO* follows as an immediate corollary to Theorem 2.

**Corollary 1.** *MLSO is a decidable set of formulas.*

The complexity of *HLSO* is generally rather high. In fact, if  $\mathcal{F}$  contains at least two function symbols, then the *HLSO*-satisfiability problem is hard for EXP-TIME. This can be proved by reducing the satisfiability problem for *attribute value logic* to *HLSO*-satisfiability. The assertion then follows since the former problem is EXPTIME-complete, by Theorem 4.5 of the paper [19].

## 5 Concluding Remarks

We proposed a rather expressive logical system for reasoning about knowledge, topology and operations on sets. We proved that the set of all theorems of this logic is finitely axiomatizable, semantically complete, and decidable. For that we took advantage of the power of hybrid logic. In this way, the most basic theoretical foundations of a tool for corresponding spatial reasoning were provided in particular.

Future research will be directed at putting this in more concrete terms. Additionally, we will utilize special operations on sets for reasoning about knowledge acquisition.

---

<sup>6</sup> This is the decisive point where the hybrid methods prove to be superior to the usual modal ones. Even if we had carried out the completeness proof for *MLSO*, we would have been forced to use the detour via hybrid logic for decidability.

## References

1. Moss, L.S., Parikh, R.: Topological reasoning and the logic of knowledge. In Moses, Y., ed.: *Theoretical Aspects of Reasoning about Knowledge (TARK 1992)*, San Francisco, CA, Morgan Kaufmann (1992) 95–105
2. Heinemann, B.: Extended canonicity of certain topological properties of set spaces. In Vardi, M., Voronkov, A., eds.: *Logic for Programming, Artificial Intelligence, and Reasoning*. Volume 2850 of *Lecture Notes in Artificial Intelligence*., Berlin, Springer (2003) 135–149
3. Heinemann, B.: A hybrid logic for reasoning about knowledge and topology (2005) Under review for journal publication. For a preliminary version see URL <http://www.informatik.fernuni-hagen.de/thi1/ber.ps>.
4. Heinemann, B.: Reasoning about knowledge and continuity. In Goebel, R., Sutcliffe, G., eds.: *Proceedings 19th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2006)*, Menlo Park, CA, AAAI Press (2006) 37–42
5. Dabrowski, A., Moss, L.S., Parikh, R.: Topological reasoning and the logic of knowledge. *Annals of Pure and Applied Logic* **78** (1996) 73–110
6. Georgatos, K.: Knowledge theoretic properties of topological spaces. In Masuch, M., Pólos, L., eds.: *Knowledge Representation and Uncertainty, Logic at Work*. Volume 808 of *Lecture Notes in Artificial Intelligence*., Springer (1994) 147–159
7. Georgatos, K.: Knowledge on treelike spaces. *Studia Logica* **59** (1997) 271–301
8. Heinemann, B.: Topological Modal Logics Satisfying Finite Chain Conditions. *Notre Dame Journal of Formal Logic* **39** (1998) 406–421
9. Weiss, M.A., Parikh, R.: Completeness of certain bimodal logics for subset spaces. *Studia Logica* **71** (2002) 1–30
10. Blackburn, P.: Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL* **8** (2000) 339–365
11. Heinemann, B.: A two-sorted hybrid logic including guarded jumps. In Schmidt, R., Pratt-Hartmann, I., Reynolds, M., Wansing, H., eds.: *Advances in Modal Logic* 5, London, King's College Publications (2005) 73–92
12. Aiello, M., Pratt-Hartmann, I., van Benthem, J.: The logic of space (2007) To appear. See URL <http://dit.unitn.it/aiellom/hsl/>.
13. Tzani, E.: Algebraizing hybrid logic. ILLC Publications MoL-2005-04, University of Amsterdam, Amsterdam (2005) See also the M4M-4-workshop programme.
14. Goldblatt, R.: *Logics of Time and Computation*. second edn. Volume 7 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford, CA (1992)
15. Chellas, B.F.: *Modal Logic*. Cambridge University Press, Cambridge (1980)
16. Aiello, M., van Benthem, J., Bezhanishvili, G.: Reasoning about space: The modal way. *Journal of Logic and Computation* **13** (2003) 889–920
17. Heinemann, B.: Topological nexttime logic. In Kracht, M., de Rijke, M., Wansing, H., Zakharyashev, M., eds.: *Advances in Modal Logic 1*. Volume 87 of *CSLI Publications*., Stanford, CA, Kluwer (1998) 99–113
18. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge (2001)
19. Blackburn, P., Spaan, E.: A modal perspective on the computational complexity of attribute value grammar. *Journal of Logic, Language and Information* **2** (1993) 129–169

# Analytic Results on the Hodgkin-Huxley Neural Network: Spikes Annihilation

Dragos Calitoiu<sup>1</sup>, John B. Oommen<sup>2</sup>, and Doron Nussbaum<sup>3</sup>

<sup>1</sup> Carleton University, Ottawa, K1S 5B6, Canada  
dcalitoi@scs.carleton.ca

<sup>2</sup> Chancellor's Professor; Fellow: IEEE and Fellow: IAPR, Carleton University,  
Ottawa, K1S 5B6, Canada  
oommen@scs.carleton.ca

<sup>3</sup> Carleton University, Ottawa, K1S 5B6, Canada  
nussbaum@scs.carleton.ca

**Abstract.** Various families of Neural Networks (NN) have been used in the study and development of the field of Artificial Intelligence (AI). More recently, the Hodgkin-Huxley (HH) has attracted a fair bit of attention. Apart from the HH neuron possessing desirable “computing” properties, it also can be used to reasonably model biological phenomena, and in particular, in modeling neurons which are “synchronized/desynchronized<sup>1</sup>”. This paper, which we believe is a of pioneering sort, derives some of the analytic/learning properties of the HH neuron, and neural network.

The HH Neuron is a nonlinear system with two equilibrium states: A fixed point and a limit cycle. Both of them co-exist and are stable. The behavior of this neuron can be switched between these two equilibria, namely *spiking* and *resting* respectively, by using a perturbation method. The change from spiking to resting is referred to as *Spike Annihilation*. In this paper, we analytically prove the existence of a brief excitation (input) which, when delivered to the HH neuron during its repetitively firing state, annihilates its spikes. We also formally derive the characteristics of this brief excitation. We thus believe that our analysis of the HH neuron has practical implications in clinical applications, especially in the case of the *desynchronization* of neuronal populations.

## 1 Introduction

In the pioneering research of Hopfield and Grossberg , the process of coding information using neural networks (NN) was developed around the regime involving fixed point attractors. An alternative philosophy, motivated by clinical neurologists, indicated that brain dynamics is characterized by cyclic and weakly chaotic regimes. Some theories proposed in Artificial Intelligence (AI) have attempted

---

<sup>1</sup> We believe that the entire concept of desynchronizing the neurons in a NN, is an area which is relatively open. Indeed, the available results concerning desynchronization are scanty.

to exploit cyclic attractors for information encoding. One of these theories consists of indexing the “attractor information items” by means of external stimuli rather than by using initial conditions as proposed by Hopfield. This algorithm implies the existence of alternative responses to external stimuli and a switching process from one of these potential attractors to another in response to any input stimulus. The process of retrieving the information stored in the cycles depends on the model chosen for the investigation: A more realistic model for the neuron will have a richer range of non-linear behaviors (represented by stable or unstable limit cycles). Our paper investigates the process of retrieving the information stored in the cycles, namely that of controlling the neuron (to be more specific, a Hodgkin-Huxley (HH) neuron).

We present now a few considerations about the dynamical properties of the HH neuron. This neural model can be in one of two states: A resting state and a state that fires in response to certain forms of stimulation. Usually, the neuron is considered to be in an equilibrium mode when it is in a resting state. However, this statement is not universal because there are two equilibrium states associated with this neuron, namely a fixed point and the limit cycle, both of which are stable. One problem to be considered here is the switching of the neuron from one equilibrium mode to the other, which is a phenomenon which can occur without modifying the number and the stability of the equilibria.

From a classical system theory point of view, the equilibrium point of a non-linear dynamical system may disappear or may lose its stability if a control parameter is changed, depending on the type of bifurcation displayed by the system. In our research, the HH neuron is considered to be a dynamical nonlinear system whose equilibrium states are not to be radically changed with regard to its stability. We investigate the case when both equilibria, namely the fixed point and the limit cycle, co-exist and remain stable. In this particular situation, the system is bi-stable, and with a carefully chosen synaptic input, it is possible to switch the behavior from being resting to one which demonstrates spiking, or from being spiking to a resting (spike annihilation) mode. The goal of this research is to describe the properties of the stimulus that can achieve this switching.

This above stimulus, chosen to be a brief pulse of current, is not a control parameter. Its behavior affects neither the existence of the equilibrium points, nor their stability. The control parameter is the strength of the constantly applied current and, during our investigation, it is set to be constant. We argue that injecting a constant current into the axon is not equivalent to injecting a brief pulse of current. In the former, the system can go through a bifurcation of the equilibrium by changing the existence of the equilibria or by affecting their stability. In the latter, however, the system can jump to an alternate location in the state space, which is achieved by the system resetting the initial condition. The neuron is driven to a state of “shock”, and consequently, the membrane potential instantly switches to a new value. The fixed point, corresponding to the resting state, co-exists with the limit cycle, which corresponds to the spiking state, and the system continues to be bistable. This leads us to the goals of

this research: (i) to prove analytically the existence of such stimuli, and (ii) to describe the characteristics of these brief depolarizing shock-stimuli that, when inserted at the appropriate time, can switch the neuron from the spiking to the resting state.

### 1.1 Contribution of the Paper

As far as we know, the entire concept of synchronization/desynchronization of the neurons has not been studied for most families because such a study not only involves the asymptotic behavior, but also an analysis of transient/periodic phenomena of the individual neurons. Without being apologetic, we mention that the paper is fairly theoretical. But all our results are supported by rigorous simulations.

As we stated before, our paper investigates only one stage of the process of retrieving the information stored in the cycles, namely that of controlling the HH neuron. In contrast to the previous pieces of work, which validated experimentally or anticipated theoretically that annihilation is possible, we achieve the followings: (i) We formally prove that the problem of spike annihilation has a well defined solution. (ii) We formally derive the characteristics of the proposed solution. All of the results are novel, and to the best of our knowledge, unknown in the field. We thus believe that our analysis of the HH neuron constitute a contribution to the process of modeling of retrieving the information and also has practical implications in clinical applications, especially in the case of the desynchronization of neuronal populations [3],[4],[5],[6].

## 2 The Bistable HH Neuron

In this section we investigate the stability-related characteristics of the HH neuron. In the previous section, we stated that the HH neuron can be perceived as a dynamical nonlinear system with two stable equilibria. This is formalized below.

Consider a two-dimensional dynamical system:

$$\frac{dV}{dt} = P(V, R), \quad (1)$$

$$\frac{dR}{dt} = Q(V, R), \quad (2)$$

where  $P(V, R)$  and  $Q(V, R)$  are polynomials of real variables  $V$  and  $R$ , and where the corresponding coefficients are real. The fundamental problem associated with the qualitative theory of such systems seems to be Hilbert's Sixteenth Problem [1], stated as follows: *Find the maximum number and the relative positions of the limit cycles of the system described by Equations (1) and (2)*. This problem remains unsolved.

It should be mentioned that there are many methods which yield *specific* results related to the study of limit cycles. However, the above general problem has not been solved, even for the simplest quadratic systems. Rather, we intend

to explore, *numerically*, the less general system defined by Equations (3) and (4) proposed by Rinzel and Wilson [2], which, indeed, approximate the Hodgkin-Huxley neuron:

$$\frac{dV}{dt} = \frac{1}{\tau}[-(a_1 + b_1V + c_1V^2)(V - d_1) - e_1R(V + f_1) + B + \sigma], \quad (3)$$

$$\frac{dR}{dt} = \frac{1}{\tau_R}(-R + a_2V + b_2), \quad (4)$$

where  $a_1, a_2, b_1, b_2, c_1, d_1, e_1, f_1, \tau$ , and  $\tau_R$  are constants,<sup>2</sup>  $B$  is the background activity,<sup>3</sup> and  $\sigma$  is an excitation stimulus. Apart from deriving certain specific analytic results, we propose to discover, *numerically*, the number and the positions of the limit cycles.

By introducing Hilbert's Sixteenth Problem as a motivation for the solutions of the system, we argue that the numerical approach to yield the number and the relative positions of the limit cycles of the system described by Equations (3) and (4), is the only reasonable strategy (instead of an analytical one) to tackle the problem.

It is true that there are some theoretical results [1], which can be postulated as theorems, that can be applied for two-dimensional nonlinear systems. But their contributions are only qualitative without being capable of describing the *complete* picture of the number and the relative positions of the limit cycles. Thus, in the interest of completeness we mention these formal results that can be used to prove that a system described by Equations (3) and (4) has a limit cycle and a bifurcation point.

## 2.1 Related Theoretical Foundation

The first useful Theorem, due to Poincare [2], states that a limit cycle must surround one or more equilibrium points.

**Poincare Theorem:** *If a limit cycle exists in an autonomous two-dimensional system, it must necessarily surround at least one equilibrium point. If it encloses exactly one equilibrium point, the latter must be a node, a spiral point, or a center, but can not be a saddle point.*

The Poincare Theorem is a necessary but not a sufficient condition for the existence of the limit cycle in systems described by Equations (3) and (4). In the general case of a two-dimensional nonlinear system, it is possible to find multiple steady states without limit cycles. Consequently, the applicability of the Poincare Theorem is limited, since by utilizing it, we will not be able to determine whether a system described by Equations (3) and (4) has a limit cycle.

<sup>2</sup> In their experiments, Rinzel and Wilson [2] set the constants as:  $a_1 = 17.81$ ,  $b_1 = 47.71$ ,  $c_1 = 32.63$ ,  $d_1 = 0.55$ ,  $e_1 = 0.55$ ,  $f_1 = 0.92$ ,  $a_2 = 1.35$ ,  $b_2 = 1.03$ ,  $\tau = 0.8$  ms and  $\tau_R = 1.9$  ms.

<sup>3</sup> The background activity generates limit cycles in the system. Without this value, the system will converge through the stable spiral point.

On the other hand, we need a theorem that specifies the conditions under which a system is forced to have a limit cycle. In the literature [2], the *Poincare-Bendixon Theorem* defines the conditions for the existence of a limit cycle. But more applicable to our scenario, is the *Hopf Bifurcation Theorem* [2], presented below, which defines the conditions for the existence of a stable or unstable limit cycle.

**Hopf Bifurcation Theorem:** Consider a nonlinear dynamical system in  $N \geq 2$  dimensions that depends on a parameter,  $\beta$ , as:

$$\frac{d\vec{X}}{dt} = \vec{F}(\vec{X}, \beta). \quad (5)$$

Let  $X_0$  be an isolated equilibrium point of the system. Assume that there is a critical value of  $\beta = \alpha$  with the following properties determined from the Jacobian,  $J(\beta)$ : (i)  $X_0$  is asymptotically stable for some finite range of values  $\beta < \alpha$ . (ii) When  $\beta = \alpha$  the system has at least one pair of pure imaginary eigenvalues  $\lambda = \pm i\omega$ , while all other eigenvalues have negative parts. (iii)  $X_0$  is unstable for some range of values  $\beta > \alpha$ .

Then, the system defined by Equation (5) either possesses an asymptotically stable limit cycle over a range  $\beta > \alpha$ , or it possesses an unstable limit cycle over some range  $\beta < \alpha$ . Furthermore, in the neighborhood of  $\beta = \alpha$ , the frequency of the oscillation characterized by the limit cycle will be approximately  $\frac{\omega}{2\pi}$ , and this oscillation will emerge with infinitesimal amplitude sufficiently close to  $\alpha$ .  $\square$

The *Hopf Bifurcation Theorem* indicates that near the critical value of  $\beta = \alpha$  there is a limit cycle. It does not tell us whether this is an unstable limit cycle occurring when  $\beta < \alpha$ , or if it is an asymptotically stable limit cycle for  $\beta > \alpha$ . However, the theorem specifies where we can search in the parameter space, to locate a limit cycle behavior. Thus, although we are not able to provide the equation that describes the limit cycle, we can identify, by investigating the neighborhood of  $\alpha$ , the qualitative description of the limit cycle.

To render our theoretical consideration meaningful, in the following, we shall derive: (i) The equilibrium states of the HH neuron by solving the system of equation described by the isoclines, (ii) The Jacobian corresponding to the system described by Equations (3) and (4), at the equilibrium states, (iii) The eigenvalues of the Jacobian, by solving the characteristic equation associated with the Jacobian, and (iv) The requirements on the eigenvalues as specified by the *Hopf Bifurcation Theorem* for identifying the limit cycle.

## 2.2 Computing the Equilibrium States

Consider a system described by Equations (3) and (4). We compute the equilibrium states by solving the system of equations described by their isoclines. This is formalized in the following Lemma.

**Lemma 1.** *The equilibrium points of the HH neuron can be obtained by solving a cubic polynomial equation:*

$$x_3 V^3 + x_2 V^2 + x_1 V + x_0 = 0 \quad (6)$$

where:  $x_3 = -c_1$ ,  $x_2 = -(b_1 + a_2e_1 - c_1d_1)$ ,  $x_1 = -(a_1 - b_1d_1 + a_2e_1f_1 + b_2e_1)$ ,  $x_0 = a_1d_1 - b_2e_1f_1 + B$ .

**Proof:** From Equations (3) and (4), we see that the system has two isoclines, specified by the contours:  $\frac{dV}{dt} = 0$  and  $\frac{dR}{dt} = 0$ , which can be written as:

$$\frac{1}{\tau}[-(a_1 + b_1V + c_1V^2)(V - d_1) - e_1R(V + f_1) + B] = 0 \quad (7)$$

and

$$\frac{1}{\tau_R}(-R + a_2V + b_2) = 0. \quad (8)$$

The background activity  $B$  is the control parameter  $\beta$  specified in the *Hopf Bifurcation Theorem*.

The equilibrium states can be computed as solutions of Equations (7) and (8). By substituting  $R$  from Equation (8) as  $R = a_2V + b_2$ , and utilizing this value in Equation (7), we obtain the equation:

$$x_3V^3 + x_2V^2 + x_1V + x_0 = 0 \quad (9)$$

where the coefficients  $x_3$ ,  $x_2$ ,  $x_1$ , and  $x_0$  are as defined in the Lemma statement. Hence the Lemma.  $\square$

### 2.3 Computing the Jacobian

We now consider a Jacobian-based analysis of the HH neuron, formalized in the following Lemma.

**Lemma 2.** *The Jacobian matrix of the system representing the HH neuron is given by:*

$$J(V, R) = \begin{pmatrix} y_{12}V^2 + y_{11}V + y_{10} & y_{21}V + y_{20} \\ y_{30} & y_{40} \end{pmatrix},$$

where  $y_{12} = -\frac{1}{\tau}3c_1$ ,  $y_{11} = -\frac{1}{\tau}(2b_1 + 2c_1d_1 + a_2e_1)$ ,  $y_{10} = -\frac{1}{\tau}(a_1 + b_1d_1 + e_1b_2)$ ,  $y_{21} = -\frac{1}{\tau}e_1$ ,  $y_{20} = -\frac{1}{\tau}f_1$ ,  $y_{30} = \frac{1}{\tau_R}a_2$ ,  $y_{40} = -\frac{1}{\tau_R}$ .

**Proof:** We know from the theory of dynamical systems that the Jacobian matrix of the system is :

$$J(V, R) = \begin{pmatrix} \frac{\partial V(V, R)}{\partial V} & \frac{\partial V(V, R)}{\partial R} \\ \frac{\partial R(V, R)}{\partial V} & \frac{\partial R(V, R)}{\partial R} \end{pmatrix}.$$

Evaluating each of these components yields:

$$\begin{aligned} \frac{\partial V(V, R)}{\partial V} &= \frac{\partial [\frac{1}{\tau}[-(a_1 + b_1V + c_1V^2)(V - d_1) - e_1R(V + f_1) + B]]}{\partial V} = \frac{1}{\tau}[-3c_1V^2 - (2b_1 + 2c_1d_1)V - (a_1 + b_1d_1) - e_1R], \\ \frac{\partial V(V, R)}{\partial R} &= \frac{\partial [\frac{1}{\tau}[-(a_1 + b_1V + c_1V^2)(V - d_1) - e_1R(V + f_1) + B]]}{\partial R} = -\frac{1}{\tau}e_1(V + f_1), \\ \frac{\partial R(V, R)}{\partial V} &= \frac{\partial (\frac{1}{\tau_R}(-R + a_2V + b_2))}{\partial V} = \frac{1}{\tau_R}a_2, \quad \frac{\partial R(V, R)}{\partial R} = \frac{\partial (\frac{1}{\tau_R}(-R + a_2V + b_2))}{\partial R} = -\frac{1}{\tau_R}. \end{aligned}$$

But Equation (8) can be used to eliminate  $R$  from the partial derivatives. By achieving this, and omitting the laborious algebraic steps, the result follows.  $\square$



## 2.4 Finding the Bifurcation Point

We shall now consider the problem of finding the neuron's bifurcation point by using the dynamical matrix of the system. This value of the bifurcation point is used to "set" the neuron so as to render it to be bi-stable.

**Theorem 1.** *A HH neuron obeying the Equations (3) and (4) has a bifurcation point if and only if a root of the equation  $\frac{1}{\tau}[-3c_1V^2 - (2b_1 + 2c_1d_1)V - (a_1 + b_1d_1) - e_1R] - \frac{1}{\tau_R} = 0$  satisfies the inequality  $V > -f_1 - \frac{1}{e_1} \frac{\tau}{\tau_R}$ .*

**Proof:** For the bifurcation point, the roots of the characteristic equation, computed from the Jacobian, are purely imaginary. It is well known that a quadratic equation  $x^2 - Sx + P = 0$  has imaginary roots if *Condition 1*:  $S = 0$ , and *Condition 2*:  $P > 0$ , where  $S$  and  $P$  are the sum and product of the roots, respectively.

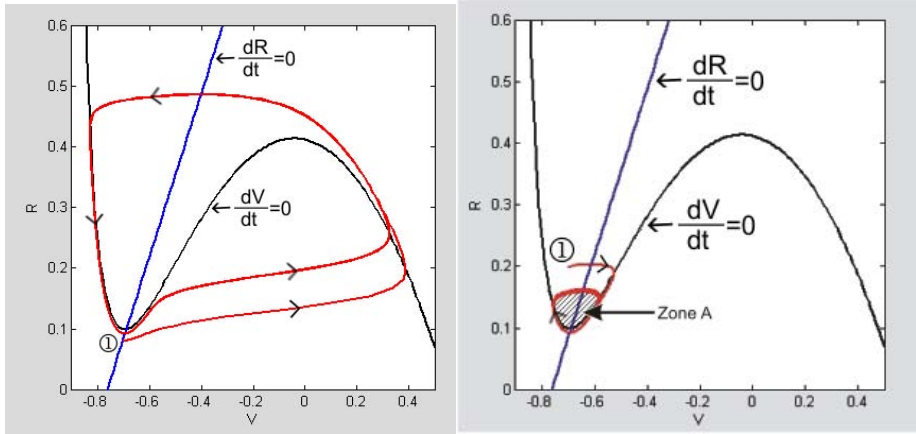
Consider the Jacobian of the HH neuron as given by *Lemma 2*. Applying *Condition 1* to this Jacobian generates the equation:  $\frac{1}{\tau}[-3c_1V^2 - (2b_1 + 2c_1d_1)V - (a_1 + b_1d_1) - e_1R] - \frac{1}{\tau_R} = 0$ . This equation has two roots, say  $V_1$  and  $V_2$ . The problem now is one of verifying whether  $V_1$  and  $V_2$  satisfy *Condition 2*. This in turn implies that for  $V_1$  and  $V_2$ :  $\frac{1}{\tau} \frac{1}{\tau_R} [-3c_1V^2 - (2b_1 + 2c_1d_1)V - (a_1 + b_1d_1) - e_1R] + \frac{1}{\tau_R} \frac{1}{\tau} e_1(V + f_1) > 0$ . We can rewrite this inequality using the observation that  $V_1$  and  $V_2$  are solutions to the equation corresponding to *Condition 1*, namely  $\frac{1}{\tau}[-3c_1V^2 - (2b_1 + 2c_1d_1)V - (a_1 + b_1d_1) - e_1R] = \frac{1}{\tau_R}$ . Using this relation, *Condition 2* becomes:  $\frac{1}{\tau_R} \frac{1}{\tau} + \frac{1}{\tau_R} \frac{1}{\tau} e_1(V + f_1) > 0$ .

We know that  $\tau_R$  and  $\tau$  are time constants, being positive. We make a convention that  $e_1$  is also a positive constant. With these considerations, *Condition 2* can be rewritten in a new form as:  $V > -f_1 - \frac{1}{e_1} \frac{\tau}{\tau_R}$ . The theorem follows since whenever these constraints are satisfied, we obtain purely imaginary roots.  $\square$

## 2.5 The Stable and Unstable Limit Cycles

If we consider  $B$  to be a control parameter, we can analytically compute the equilibrium point, which, for certain values of  $\sigma$ , leads to a *spiral stable point*, and which, for other values of  $\sigma$ , leads to an *unstable spiral point*. The behavior around a specific value, namely the change of the stability of the equilibrium point, induces the concept of a *subcritical (hard) Hopf bifurcation*. By plotting the evolutions of the numerical solutions of the system (Equations (3) and (4)), we discover that for the settings of Rinzel and Wilson [2], there is a stable limit cycle to the right of the bifurcation point. To identify a hypothetical unstable limit cycle, we can modify the system's equations to make time run "backwards". The modification, which consists of rendering the sign of the two constants,  $\tau$  and  $\tau_R$ , to be negative, changes the unstable limit cycle to become asymptotically stable. In this way, by using a numerical method, we can identify the position of a second limit cycle, which happens to be unstable. The stable spiral point is surrounded by this unstable limit cycle which, in turn, acts as a *separatrix* defining a basin of attraction for the stable point.

In Figure 1 we present the stable and unstable limit cycles, together with the isoclines ( $\frac{dV}{dt} = 0$  and  $\frac{dR}{dt} = 0$ ). The trajectory starts at the point indicated by



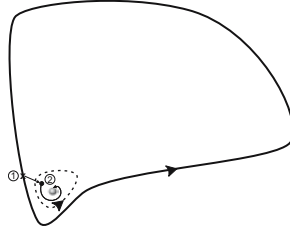
**Fig. 1.** (left side) The phase space representing the *stable* limit cycle and the resulting isoclines ( $\frac{dV}{dt} = 0$  and  $\frac{dR}{dt} = 0$ ) obtained by using Rinzel and Wilson settings for the HH neuron. The starting point, (represented with '1') is  $V_0 = -0.7$ , and  $R = 0.08$ . In addition,  $B = 0.08$ . (right side) The phase space representing the *unstable* limit cycle. The starting point must be outside the zone called  $Zone_A$ , defined by the cycle. In this graph, the starting point (represented with '1') is  $V_0 = -0.7$ , and  $R_0 = 0.2$ . In addition,  $B = 0.08$ .

'1' and follows the arrowed curves. Observe that in the case of Figure 1 (left side), the trajectory of the HH neuron trajectory follows the stable limit cycle, and in Figure 1 (right side), the trajectory follows the unstable limit cycle. When  $B$  is increased from the resting value, the steady state remains asymptotically stable and the spikes are generated only after the bifurcation point is reached, by increasing the value of  $B$ .

### 3 The Problem of Annihilation

The problem of the annihilation of spikes for the HH neuron involves moving the state of the system, by using a pulse stimulus, from outside a particular zone (denoted as  $Zone_A$ ) to being inside  $Zone_A$ , where  $Zone_A$  is a basin of attraction of the stable spiral point which is described by an unstable limit cycle. For example, if the system is characterized by the settings specified by Rinzel and Wilson [2],  $Zone_A$  is contained in the region given by  $V \in [-0.6, -0.8]$  and  $R \in [0.1, 0.15]$ , as depicted in Figure 1. Figure 2 contains all the steady states of the system, including the stable spiral point, and the stable and unstable limit cycles.

The success of the annihilation process depends on four crucial issues: (i) What should be the initial point  $(V, R)$  for the system to exhibit annihilation? (ii) When should the pulse stimulus,  $\sigma$ , be applied to the system to annihilate it? (iii) What should the amplitude of the pulse stimulus be for the annihilation



**Fig. 2.** The annihilation process for the system specified in Figure 1. The stable fixed point, the stable limit cycle, and the unstable limit cycle (the *separatrix* given by the dashed line) are represented together. If the system starts in a carefully chosen configuration at State 1 on the stable limit cycle, the system can be driven to State 2 by applying a carefully chosen stimulus. From this state, it will then go to the stable fixed point.

to be achieved? (iv) What should the duration of the pulse stimulus be for the annihilation to be achieved? The solution of the annihilation problem consists of determining a stimulus which adequately responds to all the above questions.

We now formally prove that the problem of spike annihilation is well-defined, and propose an algorithm for finding a solution to it. The problem is clarified in Figure 2. If the system starts in a carefully chosen configuration at State 1 on the stable limit cycle, the system can be driven to State 2 by applying a carefully chosen stimulus. From this state, it will then go to the stable fixed point.

We plan to analytically demonstrate that the spike annihilation problem has a well-defined solution. The strategy of solving this problem consists of: (i) Computing the steady states. (ii) Analyzing the stability of the steady states. (iii) Computing the bifurcation points and the bifurcation diagram. (iv) Computing the stable and unstable limit cycles. (v) Analyzing the existence of the stimulus that can annihilate the system.

### 3.1 The HH Neuron Annihilation Theorem

Since we are interested in annihilating the spikes, we shall demonstrate that this can be done by invoking a discretized<sup>4</sup> time model. To achieve this, first of all, we rewrite the dynamical system of equations for a bistable model of the HH neuron in a discrete-time manner as:

$$V[n+1] = V[n] + \frac{1}{\tau} [-(a_1 + b_1 V[n] + c_1 V^2[n])(V[n] - d_1) - e_1 R[n](V[n] + f_1) + Bk + \sigma], \quad (10)$$

$$R[n+1] = R[n] + \frac{1}{\tau_R} (-R[n] + a_2 V[n] + b_2). \quad (11)$$

The general **Theorem of Annihilation** is formally written below.

<sup>4</sup> A continuous-time approach cannot be invoked to prove this theorem, because, by virtue of its relation to Hilbert's Sixteenth Problem, it is not known how we can compute the explicit solutions for the system of equations.

**Theorem 2 (HH Neuron Annihilation).** Consider a system described by its discretized dynamical equations:

$$\begin{pmatrix} V[n+1] \\ R[n+1] \end{pmatrix} = \begin{pmatrix} V[n] \\ R[n] \end{pmatrix} + \begin{pmatrix} f_1(V[n], R[n]) \\ f_2(V[n], R[n]) \end{pmatrix} + \underline{S}[n], \quad (12)$$

with  $n = 0, 1, \dots$ , where  $f_1$  and  $f_2$  specify the unexcited dynamics, and  $\underline{S}[n]$  is the excitation applied to the system.

If the system has a stable limit cycle, a stable spiral point and an unstable limit cycle which separates the fixed point and the stable limit cycle, then, there exists an excitation function  $\underline{S}[n]$ , which equals 0 everywhere except at a specific point  $(V[0], R[0])$  on the stable limit cycle, at which point  $\underline{S}[0]$  has the value  $[A, 0]^T$  for a duration of one iteration, and which when applied to the system, forces it from the stable limit cycle to the stable spiral point.

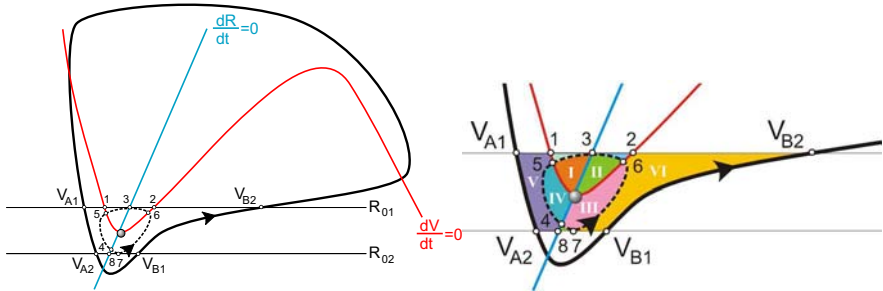
**Proof:** Consider the system defined by Equation (12), which has the excitation  $\underline{S}[n]$ . Analyzing the Jacobian of the system, we observe that it has the same form as the one corresponding to the continuous case. Thus, all the qualitative results obtained in the previous Section are also applicable for the discrete time approach, and thus, the system has a stable fixed point, a stable limit cycle and an unstable limit cycle (also known as a *separatrix*).

For the purpose of proof, we define, three distinct areas in the state space, as depicted by Figure 3: (i) We denote  $A_{In}$  as the region corresponding to the basin of attraction of the stable fixed point, bordered by the separatrix. (ii) We observe two regions outside the separatrix, that can have as their boundaries the tangents in the maximum and minimum ‘R’ points on the separatrix, the stable limit cycle and the isoclines. We denote them as:

$A_{Out,1}$ : The region where  $V[n+1] > V[n]$  and  $R[n+1] < R[n]$ , and

$A_{Out,2}$ : The region where  $V[n+1] > V[n]$  and  $R[n+1] > R[n]$ .

Let us denote the intersection between the tangents in the maximum and minimum ‘R’ points on the separatrix, and the stable limit cycle (see Figure 3) as



**Fig. 3.** (left side) The stable spiral point, the stable and the unstable limit cycle for the bi-stable HH neuron. (right side) A zoom-in of the phase space. The regions  $A_{Out,1}$  and  $A_{Out,2}$  correspond to Area V and Area VI, respectively. The regions  $A_{In,1}$ ,  $A_{In,2}$ ,  $A_{In,3}$ , and  $A_{In,4}$  correspond to Area I, Area II, Area III, and Area IV, respectively.

$V_{A1}, V_{A2}, V_{B1}, V_{B2}$ . The sequence of these points corresponds to the time evolution on the stable limit cycle.

Within the discrete-time model of computation, the problem of annihilation involves proving that there exists a stimulus  $A$ , which when applied between  $V_{A1}$  and  $V_{A2}$  or between  $V_{B1}$  and  $V_{B2}$  moves the system into the basin of attraction of the stable fixed point, namely within  $A_{In}$ . Observe that if the system is within this region, it is inside the separatrix, and it will thus converge to the fixed point. Indeed, it suffices to show that this input can be applied for a single time unit.

Consider the scenario in which the system is on an initial point  $V[0]$  between  $V_{A1}$  and  $V_{A2}$ . Since the stable limit cycle and the separatrix are non-intersecting, there exists a positive “distance”  $d_0$  between  $V[0]$  and the separatrix. We intend to determine a value of  $A$  that moves the system from  $(V[0], R[0])$  to an arbitrary point in  $A_{In}$ . Clearly, the magnitude  $A$  has to satisfy the condition :

$$(V[1] - V[0]) > d_0 \quad (13)$$

Computing  $V[1]$  as a function of  $V[0]$  we have:  $V[1] = V[0] + f_1(V[0]) + A$ . The condition (13) becomes:

$$(V[0] + f_1(V[0] + A - V[0]) > d_0 \implies (f_1(V[0] + A) > d_0 \implies A > d_0 - f_1(V[0])). \quad (14)$$

We now invoke the monotonic property of the function  $V[n]$ , that corresponds to the portion of the state space below the isocline, where  $V[n+1] > V[n]$ , namely in  $A_{In}$ . Here, the term  $f_1(V[n]) = V[n+1] - V[n]$  is positive. We thus see that there exists a value of  $A$ , satisfying the condition (14), that moves the initial point of the system between  $V_{A1}$  and  $V_{A2}$ , to be in  $A_{In}$ . We have now to evaluate the sign of the expression  $[d_0 - f_1(V[0])]$ . Starting from  $(V[0], R[0])$  on the stable limit cycle, with  $V[0]$  between  $V_{A1}$  and  $V_{A2}$ , we know that, without adding the  $A$  stimulus, the next point  $(V[1], R[1])$  will also be on the stable limit cycle. The difference between  $V[1]$  and  $V[0]$  is exactly  $f_1(V[0])$ . In this context,  $f_1(V[0])$  will satisfy the condition  $f_1(V[0]) < d_0$ , because there is no intersection, between the limit cycle and the unstable limit cycle (described by the separatrix). We have now thus proved that  $[d_0 - f_1(V[0])] > 0$ . Thus,  $A$  is a positive value satisfying  $A > d_0 - f_1(V[0])$ .

The analogous rationale can be used if the initial point  $V[0]$  is between  $V_{B1}$  and  $V_{B2}$ . In this case, there exists a distance  $d_1$  (a positive value) between  $V[0]$  and the separatrix. We intend again to find a value of  $A$  that moves the system into region  $A_{In}$ . The magnitude that  $A$  has to satisfy, leads to the condition :

$$(V[0] - V[1]) > d_1. \quad (15)$$

Observe also that this part of the state space, (also below the isocline), corresponds to  $V[n+1] > V[n]$ , and, thus, the term  $f_1(V[n]) = V[n+1] - V[n]$  is also positive.

Computing  $V[1]$  and  $R[1]$  as a function of  $V[0]$  and  $R[0]$  we have:  $V[1] = V[0] + f_1(V[0]) + A$ . The condition (15) thus becomes:

$$(V[0] - V[0] - f_1(V[0]) - A > d_1 \implies A < -d_1 - f_1(V[0])). \quad (16)$$

Observe that both  $d_1$  and  $f_1(V[0])$  are positive quantities, and the term  $[-d_1 - f_1(V[1])]$  is a negative value. We have proved that there exists a value of  $A$  that moves the initial point of the system from being between  $V_{B1}$  and  $V_{B2}$ , to be within  $A_{In}$ . Since both these cases are exhaustive, the theorem is proved.  $\square$

## 4 Conclusions

The literature about the synchronization/desynchronization of NNs is scanty because such an analysis would involve transient/periodic phenomena of the individual neurons. We have made a small step in this regard to analyze the synchronization properties of one such NN, the network of HH neurons. This paper briefly described the HH neuron and formally derived various properties of its stability. It also described the formal proof that the problem of spike annihilation has a well defined solution, and presented an algorithm for computing the properties of the stimulus. We add that the method of perturbation with brief stimuli differs from the classical approach of modifying the control parameter and changing the Jacobian of the system. In our approach, we keep the system bi-stable all the time, and our task is to switch between these two states without modifying their stability. To conclude, we have analytically proved the existence of the brief current pulse that annihilates the spikes of the HH neuron.

## References

1. Gray, J.J.: The Hilbert Challenge. Oxford University Press (2000)
2. Wilson, H.: Spikes decisions and actions: Dynamical foundations of neuroscience. Oxford University Press (1999)
3. Mayberg, H.S., Lozano, A.M., Voon, V., McNeely, H.E., Seminowicz, D., C Hamani J. M. Schwalb, S.H.K.: Deep brain stimulation for treatment-resistant depression. *Neuron* **45**, 651–660 (2005)
4. R. Guttman, S.L., Rinzel, J.: Control of repetitive firing in squid axon membrane as a model for a neurooscillator. *Journal of Physiology* **305**, 377–395 (1980)
5. Rinzel, J.: Numerical calculation of stable and unstable periodic solutions to the Hodgkin-Huxley equations. *Mathematical Biosciences* **49**, 27–59 (1980)
6. T. Teorell: A biophysical analysis of mechano-electrical transduction. in *Handbook of Sensory Physiology*, vol 1, (W.R. Loewenstein ed.), Springer Verlag, Berlin pp. 291–339 (1971)

# Improving Importance Sampling by Adaptive Split-Rejection Control in Bayesian Networks

Changhe Yuan<sup>1</sup> and Marek J. Druzdzel<sup>2</sup>

<sup>1</sup> Mississippi State University, Mississippi State, MS 39762, USA  
cyuan@cse.msstate.edu

<sup>2</sup> School of Information Sciences and Intelligent Systems Program,  
University of Pittsburgh, Pittsburgh, PA 15260, USA  
marek@sis.pitt.edu

**Abstract.** Importance sampling-based algorithms are a popular alternative when Bayesian network models are too large or too complex for exact algorithms. However, importance sampling is sensitive to the quality of the importance function. A bad importance function often leads to much oscillation in the sample weights, and, hence, poor estimation of the posterior probability distribution. To address this problem, we propose the *adaptive split-rejection control* technique to adjust the samples with extremely large or extremely small weights, which contribute most to the variance of an importance sampling estimator. Our results show that when we adopt this technique in the EPIS-BN algorithm [14], adaptive split-rejection control helps to achieve significantly better results.

## 1 Introduction

Bayesian networks [11] have become core tools for knowledge representation in Artificial Intelligence. One obstacle to their application is their computational complexity, which is NP-hard [3]. Importance sampling based algorithms are a popular alternative when Bayesian network models are too large or too complex for exact algorithms. The state of the art importance sampling algorithm is the EPIS-BN algorithm [14], whose main idea is to use several steps of *loopy belief propagation* (LBP) [10] to estimate an importance function for importance sampling. The algorithm is shown to reach the precision limit of sampling algorithms on some networks. However, due to the potential instability of LBP and, hence, possibly poor importance functions, EPIS-BN can still perform sub-optimally. This is manifested in the oscillation in the sample weights. Samples with extremely large weights and extremely small weights contribute most to the variance of the estimator. To address this problem, we propose an *adaptive split-rejection control* technique to adjust these samples. The technique consists of two parts: *adaptive split control*, which is to adaptively split samples with extremely large weights, and *adaptive rejection control*, which is to stochastically reject samples with extremely small weights. Although the adaptive split-rejection control may introduce correlation into the samples, the correlation is minimal in comparison to the reduction in the overall variance for sample sets with much oscillation. Our results show that when we apply adaptive split-rejection control in EPIS-BN, it helps the algorithm to achieve significantly better results. Two closely related techniques are

*resampling* [9] and *pruned-enriched Rosenbluth method* (PERM) [5,7,12]. The major advantage of our technique is that it exploits the fact that importance sampling in Bayesian networks is a high dimensional problem such that it generates useful samples more efficiently.

## 2 Importance Sampling

We start with the theoretical roots of importance sampling. We will use capital letters for variables and lowercase letters for their states. Bold letters denote sets of variables or states. Let  $f(\mathbf{X})$  be a function of  $n$  variables  $\mathbf{X} = \{X_1, \dots, X_n\}$  over the domain  $\Omega \subset R^n$ . Consider the problem of estimating the multiple integral

$$V = \int_{\Omega} f(\mathbf{x}) d\mathbf{x} . \quad (1)$$

We assume that the domain of integration of  $f(\mathbf{X})$  is bounded, i.e., that  $V$  exists. Importance sampling approaches this problem by estimating

$$V = \int_{\Omega} \frac{f(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} , \quad (2)$$

where  $g(\mathbf{X})$ , called the *importance function*, is a probability density function such that  $g(\mathbf{X}) > 0$  across the entire domain  $\Omega$ . One practical requirement of  $g(\mathbf{X})$  is that it should be easy to sample from. In order to estimate the integral, we generate samples  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$  from  $g(\mathbf{X})$  and use the generated values in the sample-mean formula

$$\hat{V} = \frac{1}{N} \sum_{i=1}^N \frac{f(\mathbf{x}_i)}{g(\mathbf{x}_i)} . \quad (3)$$

The estimator almost surely converges to  $V$  under certain weak assumptions [4].

The performance of the estimator in Equation 3 can be measured by its variance

$$\text{var}_g\left[\frac{f(\mathbf{X})}{g(\mathbf{X})}\right] = \int_{\Omega} \frac{f^2(\mathbf{x})}{g(\mathbf{x})} d\mathbf{x} - V^2 . \quad (4)$$

Rubinstein [13] shows that if  $f(\mathbf{X}) > 0$ , the optimal importance function is

$$g(\mathbf{X}) = \frac{f(\mathbf{X})}{V} . \quad (5)$$

In this case, the variance of the estimator is zero. However, the concept of the optimal importance function is of rather theoretical significance because finding  $V$  is equivalent to finding the posterior distribution, which is the problem that we are facing. Nevertheless, it suggests that if we find instead a function that is close enough to the optimal importance function, we can still expect good convergence rates.



### 3 Improving Importance Sampling by Adaptive Split-Rejection Control

Since we normally have no access to the optimal importance functions, we have to resort to approximation techniques or even guess work to estimate them. Importance functions thus obtained may yield far from optimal performance, which is directly reflected in the much oscillation in the sample weights. In such a sample set, extremely large weights and extremely small weights contribute most to the variance of the estimator. Furthermore, if the ratio  $f/g$  is unbounded, the variance of the estimator may be infinite, in which case importance sampling is unreliable. To remedy this problem, we propose the *adaptive Split-Rejection control* technique to adjust these samples. The technique includes two parts: *adaptive split control* and adaptive rejection control.

#### 3.1 Adaptive Split Control

Samples with extremely large weights have a large impact on the variance of an importance sampling estimator, because they dominate the estimator and make other samples less useful. If some of these large-weight samples are in the unimportant parts of  $f$ , the estimator will wrongly put too much emphasis on these parts, and its performance will inevitably be poor. To deal with this problem, we propose the *adaptive split control* technique. The basic idea is to split a sample with a large weight into several samples with smaller weights. This technique is related to the enrichment method in [12,5,7]. In Bayesian networks, since we need to go over all the nodes in the topological order to draw a single sample, we can further exploit the power of adaptive split control by appointing several rejection nodes, say every 50th nodes. More formally, adaptive split control works as follows.

Adaptive Split Control:

1. For  $i = 1, \dots, M$ , draw  $\mathbf{x}_i$  from  $g(\mathbf{X})$ , and keep track of the cumulative weights for every rejection node.
2. For all the rejection nodes, sort the weights ascendingly, and let split threshold  $c_s = w_{\lfloor \alpha_s \times M \rfloor}$  using the choose *split percentile*  $\alpha_s$ .
3. For  $j = 1, \dots, N$  ( $N \gg M$ ), draw  $\mathbf{x}_j$  from  $g(\mathbf{X})$  according to Steps 4 and 5.
4. Go over each node in the topological order of the network, and instantiate it to a state that is sampled from its distribution conditional on its parents.
5. If the cumulative weight  $w$  becomes larger than the split threshold  $c_s$  at a rejection node, we split it into  $k = \lfloor w/c_s + 1 \rfloor$  samples, each with weight  $w/k$ .

It is easy to show that the new estimator is still unbiased.

**Theorem 1.** *The importance sampling estimator with adaptive split control is unbiased.*

There are two purposes for the adaptive split control technique. First, we can prevent a sample weight from blowing up and get samples with more uniform weights. Second, after we split the sample, the resulting samples enable us to explore a wider range of the sample space, and we are more likely to get better samples. Intuitively, adaptive

split control tries to modify the old importance function such that the new importance function  $g_s(\mathbf{X})$  approximately follows the following distribution:

$$g_s(\mathbf{X}) = q_{c_s}^{-1} \max\{g(\mathbf{X}), \frac{f(\mathbf{X})}{c_s}\}, \quad (6)$$

where  $q_{c_s}$  is defined as

$$q_{c_s} = \int_{\Omega} \max\{1, \frac{w(\mathbf{x})}{c_s}\} g(\mathbf{x}) d\mathbf{x}, \quad (7)$$

where  $w(x) = f(x)/g(x)$ . The result is only approximate because of the way that we calculate the number of split samples. Suppose the target density and the importance function are both normalized, the following theorem shows that adaptive split control reduces the  $\chi^2$  distance between the two distributions, which is defined as

$$\chi^2(f, g) = \int_{\Omega} \frac{[f(\mathbf{x}) - g(\mathbf{x})]^2}{g(\mathbf{x})} d\mathbf{x} \equiv \text{var}_g\left[\frac{f(\mathbf{X})}{g(\mathbf{X})}\right]. \quad (8)$$

**Theorem 2.** *The  $\chi^2$  distance between the modified importance function in Equation 6 and the target density is smaller than that between the original importance function and the target density; that is,*

$$\text{var}_{g_s}\left[\frac{f(\mathbf{X})}{g_s(\mathbf{X})}\right] \leq \text{var}_g\left[\frac{f(\mathbf{X})}{g(\mathbf{X})}\right]. \quad (9)$$

*Proof.* Let

$$h(w_1, w_2) = [\max\{w_1, c_s\} - \max\{w_2, c_s\}] \times [w_1 \min\{w_1, c_s\} - w_2 \min\{w_2, c_s\}].$$

$h(w_1, w_2)$  is always nonnegative, because

$$h = \begin{cases} 0 \geq 0, & \max(w_1, w_2) < c_s, \\ c_s(w_1 - w_2)^2 \geq 0, & c_s < \min\{w_1, w_2\}, \\ (c_s - w_2)(w_1^2 - w_2 c_s) \geq 0, & w_1 \leq c_s \leq w_2, \\ (w_1 - c_s)(w_1 c_s - w_2^2) \geq 0, & w_2 \leq c_s \leq w_1. \end{cases}$$

Hence,  $\max\{w(\mathbf{x}), c_s\}$  and  $\min\{w(\mathbf{x}), c_s\}w(\mathbf{x})$  are positively correlated. Therefore,

$$\begin{aligned} & c_s[1 + \text{var}_{g_s}\left\{\frac{f(\mathbf{x})}{g_s(\mathbf{x})}\right\}] \\ &= q_{c_s} E_g[\min\{w(\mathbf{x}), c_s\}w(\mathbf{x})] \\ &= E_g[\min\{w(\mathbf{x}), c_s\}w(\mathbf{x})] E_g \max\{c_s, w(\mathbf{x})\} \\ &\leq E_g[\{\min\{w(\mathbf{x}), c_s\} \max\{c_s, w(\mathbf{x})\}w(\mathbf{x})\}] \\ &= c_s E_g[w^2(\mathbf{x})] \\ &= c_s[1 + \text{var}_g\left\{\frac{f(\mathbf{x})}{g(\mathbf{x})}\right\}]. \end{aligned}$$

Although splitting samples may introduce correlations among the samples, we can look at it in a different way. After we increase the mass of the importance function in the parts where large-weight samples come from, these weights will have smaller weights. In the mean time, we also have more chances to hit these samples. In some sense, we are performing selective resampling; we are only resampling from samples with extreme weights. Also, there is an interesting connection between adaptive split control and *Rao-Blackwellization* [8]. The idea of Rao-Blackwellization is, if we can perform part of the multiple integral in Equation 1 analytically, we can reduce the variance of the estimator in Equation 3. In adaptive split control, instead of performing exact integration, we draw multiple samples to estimate that part of the integral, which can also reduce the variance of the estimator.

### 3.2 Adaptive Rejection Control

Samples with extremely large weights have a large impact on the variance of an importance sampling estimator. Similarly, samples with extremely small weights also have impact on the variance. Since they are very small, they do not play much role in the estimator but only make its variance larger. Simply throwing them away is not good because that introduces bias. To adjust these samples, we can apply a technique called *rejection control* (RC) [8]. Suppose we have drawn samples  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$  from  $g(\mathbf{X})$ . Let  $w_j = f(\mathbf{X}_j)/g(\mathbf{X}_j)$ . Rejection control conducts the following operation for any given threshold value  $c_r > 0$ .

Rejection Control:

1. For  $j = 1, \dots, N$ , accept  $\mathbf{x}_j$  with probability

$$r_j = \min\left\{1, \frac{w_j}{c_r}\right\}. \quad (10)$$

2. If the  $j$ th sample  $\mathbf{x}_j$  is accepted, its weight is updated to  $w_{*j} = q_{c_r} w_j / r_j$ , where

$$q_{c_r} = \int \min\left\{1, \frac{w(\mathbf{x})}{c_r}\right\} g(\mathbf{x}) d\mathbf{x}. \quad (11)$$

It can also be easily shown that rejection control is an unbiased operation.

**Theorem 3.** *The importance sampling estimator with rejection control is unbiased.*

Similarly to adaptive split control, rejection control adjusts the importance function so that the new importance function  $g_r(\mathbf{X})$  is expected to be closer to the target function  $f(\mathbf{X})$ . In fact, it is easily seen that

$$g_{c_r}(\mathbf{X}) = q_{c_r}^{-1} \min\left\{g(\mathbf{X}), \frac{f(\mathbf{X})}{c_r}\right\}. \quad (12)$$

The following theorem can be proven analogously to Theorem 2.

**Theorem 4.** [8] *The  $\chi^2$  distance between the target density and the modified importance function in Eqn. 12 is smaller than that between the target density and the original importance function; that is,*

$$\text{var}_{g_r}\left[\frac{f(\mathbf{X})}{g_r(\mathbf{X})}\right] \leq \text{var}_g\left[\frac{f(\mathbf{X})}{g(\mathbf{X})}\right]. \quad (13)$$

Theorem 4 is a very strong result, because it holds essentially for any  $c_r$ . However, it has also been shown that rejection control can be looked on as an importance sampling algorithm in a higher dimension, and the  $\chi^2$  distance between the target density and the modified importance function for the new algorithm is larger than that of the importance sampling algorithm without rejecting any samples [1].

To make rejection control useful, we propose the *adaptive rejection control* technique, which extends rejection control in two ways. First, to draw one sample for a Bayesian network, we need to go over all the nodes in the network. However, we need not wait until the end in order to figure out that some samples may have extremely low weights. If we find that they become very small and not promising before we finish drawing them, we can reject them early. Therefore, the first extension works as follows.

- *Early rejection:* Instead of using rejection control after we get a complete sample, we apply this technique for each, say 50th, node when drawing each sample.

This extension reduces the cost of rejecting samples and improves the efficiency in obtaining effective samples; It is the key element that makes rejection control useful.

Second, we notice that it is hard to specify a  $c_r$  in advance when we apply rejection control to importance sampling in Bayesian networks, because the posterior distribution is only known up to a constant. Therefore, the second extension is:

- *Adaptive rejection:* Instead of setting a rejection threshold  $c_r$  in advance, we draw an initial sample set with size  $M$ , sort the sample weights  $w_i$ s, and let  $c_r = w_{\lfloor \alpha \times M \rfloor}$ , where  $\alpha$  is a chosen *rejection percentile*.

The bigger  $\alpha$  is, the more samples are likely to be rejected, so we need to draw more samples in order to obtain a predefined number of samples. The cost is that the running time of the algorithm will inevitably increase. Therefore, we can fine-tune the choice of  $\alpha$  in order to achieve a satisfactory tradeoff between precision and efficiency.

To summarize, the adaptive rejection control technique works as follows:

Adaptive Rejection Control:

1. For  $i = 1, \dots, M$ , draw  $\mathbf{x}_i$  from  $g(\mathbf{X})$ , and keep track of the cumulative weights for every 50th node.
2. Sort the weights ascendingly, and let  $c_r = w_{\lfloor \alpha_r \times M \rfloor}$  using the choose *rejection percentile*  $\alpha_r$  for all the rejection nodes.
3. For  $j = 1, \dots, N$  ( $N \gg M$ ), draw  $\mathbf{x}_j$  from  $g(\mathbf{X})$ , during which we apply rejection control technique each time when we come across a rejection node.

### 3.3 Adaptive Split-Rejection Control

We call the resulting technique by putting adaptive split control and adaptive rejection control together *adaptive split-rejection control*. The following corollary follows from Theorem 1 and 3.

**Corollary 1.** *The importance sampling estimator with adaptive split-rejection control is unbiased.*

**Algorithm:** ASREPIS-BN**Input:** Bayesian network  $B$  with variables  $\mathbf{X}$ , a set of evidence variables  $\mathbf{E}$ ;**Output:** The marginal distributions of non-evidence variables.

1. Order the nodes in their topological order.
2. Initialize parameters  $\alpha_s, \alpha_r, M, N$ , and  $\epsilon$ .
3. Use several steps of LBP to calculate an importance function.
4. Enhance the importance function by  $\epsilon$ -cutoff.
5. **for**  $j \leftarrow 1$  **to**  $M$  **do**
  - for** each  $X_j$  in  $\mathbf{X}$ 
    - Sample  $\mathbf{x}_j$  according to  $P(X_j | \text{PA}(X_j))$ .
    - Calculate the partial score  $w_{jScore}$ .
    - if**  $j \% 50 == 0$ , store  $w_{jScore}$ .
  - end for**
6. Calculate thresholds  $c_s$  and  $c_r$  for all rejection nodes.
7. Set  $i = 0$ .
8. **while**  $i < N$  **do**
  - for** each  $X_j$  in  $\mathbf{X}$ 
    - Sample  $\mathbf{x}_j$  according to  $P(X_j | \text{PA}(X_j))$ .
    - Calculate the partial score  $w_{jScore}$ .
    - if**  $j \% 50 == 0$  and  $w_{jScore} < c_r^j$ 
      - Accept  $\mathbf{x}_j$  with  $p = w_{jScore} / c_r^j$ .
      - if** accepted,  $w_{jScore} = c_r^j$ ; **else**,  $w_{jScore} = 0$ .
    - else if**  $j \% 50 == 0$  and  $w_{jScore} > c_s^j$ 
      - Split  $\mathbf{x}_j$  into  $k = w_{jScore} / c_s^j$  samples.
      - Assign each sample weight  $w_{jScore} / k$ .
    - end if**
  - end for**
  - if**  $w_{iScore} > 0$ 
    - Add  $w_{iScore}$  to the score tables.
    - $i \leftarrow i + 1$ ;
  - end if**
9. Normalize each score table, and output the estimated beliefs for each node.

**Fig. 1.** The Adaptive Split-Rejection Controlled Evidence Pre-propagation Importance Sampling (ASREPIS-BN) Algorithm

Furthermore, it is easily seen that the new importance function of the adaptive split-rejection control has the following form.

$$g_{c_s r}(\mathbf{X}) = (q_{c_s} q_{c_r})^{-1} \min \left\{ \max \left\{ g(\mathbf{X}), \frac{f(\mathbf{X})}{c_s} \right\}, \frac{q_{c_s} f(\mathbf{X})}{c_r} \right\}. \quad (14)$$

The following corollary immediately follows from Theorems 2 and 4.

**Corollary 2.** *The  $\chi^2$  distance between the target density and the modified importance function in Eqn. 14 is smaller than that between the target density and the original importance function; that is,*

$$\text{var}_{g_{sr}}\left[\frac{f(\mathbf{X})}{g_{sr}(\mathbf{X})}\right] \leq \text{var}_g\left[\frac{f(\mathbf{X})}{g(\mathbf{X})}\right]. \quad (15)$$

This technique not only allows us to discard some poor samples well before we finish drawing them, but also enables us to explore a wider range of the sample space. The technique is very general, because it can be applied in any existing importance sampling algorithm for Bayesian networks.

However, we should not blindly apply adaptive split-rejection control the same way under all circumstances. When the original sample set is already good, the correlation introduced by the technique may become dominant in comparison to the reduction of the sample variance, so the results may become worse. In that case, it is desirable to adjust the split and rejection percentiles or simply switch it off altogether in order to reduce the number of samples involved in the control. The quality of a sample set can be evaluated by the *coefficient of variation* ( $cv^2(w)$ ) [8]. The  $cv^2(w)$  of the unnormalized weights is defined as follows:

$$cv^2(w) = \frac{\sum_{j=1}^N (w(\mathbf{x}_j) - \bar{w})^2}{(N-1)\bar{w}^2}. \quad (16)$$

$cv^2(w)$  is a good estimator of the  $\chi^2$  distance between the importance function and the target distribution. We recommend to increase the split percentile and the decrease rejection percentile as  $cv^2(w)$  decreases, and we switch off the adaptive split-rejection control as  $cv^2(w)$  drops below some threshold value  $cv_c$ .

We extend the EPIS-BN algorithm using the above adaptive rejection control technique, which results in the *Adaptive Split-Rejection Controlled Evidence Pre-propagation Importance Sampling* (ASREPIS-BN) algorithm, outlined in Figure 1. The algorithm has three main stages. The first stage (Steps 1-4) applies loopy belief propagation and  $\epsilon$ -cutoff to calculate an importance function. The second stage (Steps 5-6) draws an initial sample set to estimate the split thresholds  $c_s$  and rejection thresholds  $c_r$ . The third stage (Steps 7-9) applies adaptive split-rejection control to do importance sampling.

## 4 Experimental Results

To test the performance of the ASREPIS-BN algorithm, we compared it against the EPIS-BN algorithm. We did experiments on the ANDES, CPCS, and PATHFINDER networks. Our comparison was based on *Hellinger's distance* [6] between exact answers and sampling results. Hellinger's distance yields results identical to Kullback-Leibler divergence in most cases, but its major advantage is that it can handle zero probabilities, which are not uncommon in Bayesian networks. We implemented our algorithm in C++ and performed our tests on a 2.8 GHz Xeon Windows XP computer with 2GB memory.

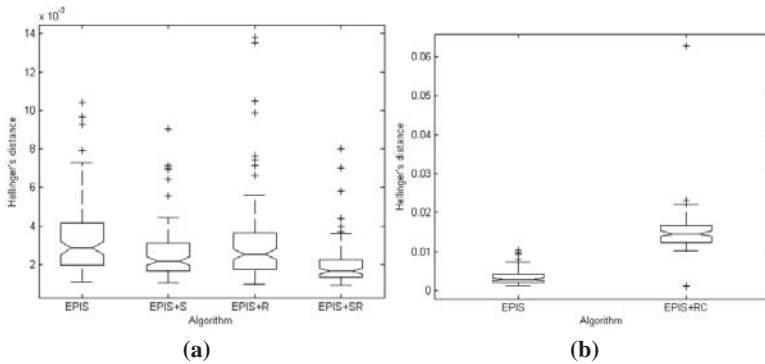
### 4.1 Review of the Performance of EPIS

Experimental results in [14,15] show that the EPIS-BN algorithm achieves a considerable improvement over the previous state of the art algorithm, the AIS-BN [2].

Furthermore, the results in [14,15] also show that the EPIS-BN algorithm already approaches the limit that sampling algorithms can achieve on CPCS and PATHFINDER, because the precision that it achieves on these networks is already in the same order as those of probabilistic logic sampling on the same networks without evidence. In the latter case, since there is no evidence in the networks, logic sampling samples from the optimal importance function, the prior distribution. We believe that precision so achieved is the limit of sampling algorithms.

### 4.2 Results of Proposed Heuristics on ANDES Network

In this experiment, we generated a total of 75 test cases for the ANDES network. These cases consisted of five sequences of 15 cases each. For each sequence, we randomly chose a different number of evidence nodes: 15, 20, 25, 30, 35 respectively. We set  $\alpha_r$  to be 0.8 and  $\alpha_s$  to be 0.99. We switched off adaptive split-rejection control when the  $cv^2(w)$  of the initial sample set was less than  $cv_c = 3.0$ . Since adaptive split-rejection control has two heuristics: adaptive split control (S) and adaptive rejection control (R), we performed experiments that aimed at disambiguating their roles. We denote EPIS-BN without any heuristic method as the EPIS algorithm, EPIS-BN with only adaptive split control as EPIS+S, EPIS-BN with only adaptive rejection control as EPIS+R, and EPIS-BN with both heuristics as EPIS+SR (ASREPIS-BN). We also tested the EPIS-BN with pure rejection control method proposed in [8], which is denoted as EPIS+RC. Again, the difference between our proposed adaptive rejection control and pure rejection control is that the former method applies rejection control periodically before finishing a complete sample, while the latter applies rejection control to complete samples. We compared the performance of EPIS, EPIS+S, EPIS+R, EPIS+SR and EPIS+RC and tested them on the same 75 test cases generated as described above. In order to be fair for all the algorithms, we let EPIS-BN run for 320K samples and let



**Fig. 2.** (a) Results of our proposed heuristics, adaptive split control (S) and adaptive rejection control (R), when applied in the EPIS-BN algorithm for a fixed running time (same as a) on ANDES network; (b) The results of EPIS-BN and EPIS-BN with pure rejection control for a fixed running time (around 12s); Pure rejection control made the performance of EPIS-BN worse on ANDES network

other algorithms run for the same amount of time. The size of the initial sample sets of EPIS+S, EPIS+R, EPIS+SR, and EPIS+RC are all  $4K$ . Boxplots of the errors of the algorithms are shown in Figure 2.

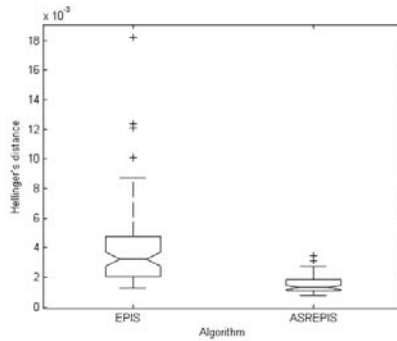
From the Figure 2b, we can see that the pure rejection control [8] made the performance of EPIS-BN worse, which is rightly pointed out in [1]. However, our proposed adaptive rejection control with several extensions, helped EPIS-BN achieve a better precision as Figure 2a shows. Figure 2a also shows that adaptive split control also helped EPIS-BN achieve a better precision. More dramatically, the two heuristics amplified each other in EPIS+SR (ASREPIS-BN) and helped it achieve a much better precision. The median errors for EPIS-BN, EPIS+S, EPIS+R and EPIS+SR are 0.0029, 0.0022, 0.0025, and 0.0017 respectively. A paired one-tail t-test indicates that the improvement of ASREPIS-BN over EPIS-BN is significant at  $p = 4.44 \times 10^{-18}$  level. We would like to stress that the improvement is achieved within the same time and with fewer samples. The ASREPIS-BN algorithm only generated, on the average,  $170K$  samples. Therefore, we conclude that the quality of the samples generated by ASREPIS-BN is much better than that of EPIS-BN.

### 4.3 Fixed Number of Samples

In this experiment, we let both EPIS-BN and ASREPIS-BN run for  $320K$  samples. Obviously, since we let ASREPIS-BN run for more samples than the last experiment, we can get even bigger improvement. Figure 3(a) shows the overall error of the results of the two algorithms. The median errors were 0.0032 for EPIS-BN and 0.0013 for ASREPIS-BN. ASREPIS-BN achieves an overall error that is less than half of that of EPIS-BN.

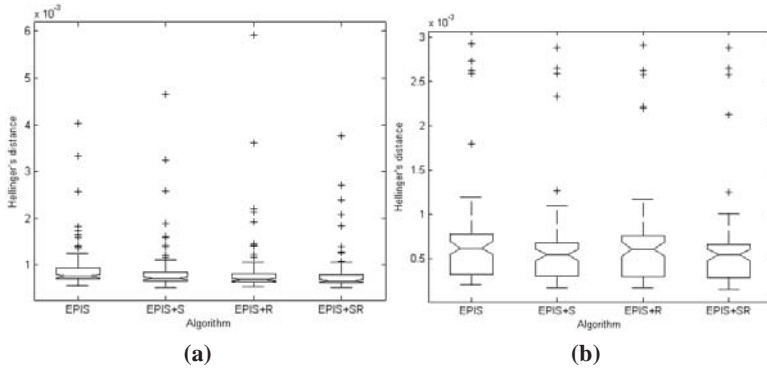
### 4.4 Results on CPCS and PATHFINDER

Since EPIS-BN already performs almost optimally on the CPCS and PATHFINDER networks, we used more conservative parameters in this experiment; We let rejection percentile  $\alpha_r$  to be 0.2, split percentile  $\alpha_s$  to be 0.999, and switch-off threshold  $cv_c$  to



**Fig. 3.** Hellinger's distance of the EPIS-BN and ASREPIS-BN algorithms on ANDES network given a fixed number of samples (320K)





**Fig. 4.** Results of our proposed heuristics, adaptive split control (S) and adaptive rejection control (R), when applied in the EPIS-BN algorithm for a fixed running time on (a) CPCS and (b) PATHFINDER

be 3.0. The results are shown in Figure 4. We can see that adaptive split-rejection control also brings some improvements for CPCS and PATHFINDER. The median errors for EPIS-BN, EPIS+S, EPIS+R, and EPIS+SR were 0.00076, 0.00072, 0.00069, 0.0006 on CPCS and 0.00063, 0.00055, 0.00061, 0.00055 on PATHFINDER respectively. Although the improvement is not much in comparison to that on ANDES, we again believe that this is due to ceiling effect: EPIS-BN already does very well on the two networks.

## 5 Conclusion

We propose the *adaptive split-rejection control* technique to stochastically reject samples with extremely small weights and adaptively split samples with extremely large weights for importance sampling in Bayesian networks. Our results show that the technique significantly improves the performance of EPIS-BN. Although adaptive split-rejection control may introduce correlation among the samples, the correlation is minimal in comparison to the reduction in the overall sample variance for sample sets with much oscillation. However, how to choose optimal split and rejection percentiles under different circumstances needs further exploration.

## Acknowledgement

This work was partly done when the first author was at University of Pittsburgh and supported by the Air Force Office of Scientific Research grant FA9550-06-1-0243.

## References

1. Y. Chen. Another look at rejection sampling through importance sampling. Technical report, Institute of Statistics and Decision Sciences, Duke University, 2004.
2. J. Cheng and M. J. Druzdzel. BN-AIS: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research*, 13:155–188, 2000.

3. G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, Mar. 1990.
4. J. Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57(6):1317–1339, 1989.
5. P. Grassberger. Pruned-enriched Rosenbluth method: Simulations of  $\theta$  polymers of chain length up to 1 000 000. *Physical Review E*, 56:3682–3693, Sept. 1997.
6. G. Kokolakis and P. Nanopoulos. Bayesian multivariate micro-aggregation under the Hellinger’s distance criterion. *Research in official statistics*, 4(1):117–126, 2001.
7. F. Liang. Dynamically weighted importance sampling in monte carlo computation. *Journal of the American Statistical Association*, 97:807–821, 2002.
8. J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, New York, 2001.
9. J. S. Liu, R. Chen, and W. H. Wong. Rejection control and sequential importance sampling. *Journal of the American Statistical Association*, 93(443):1022–1031, 1998.
10. K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 467–475, San Francisco, CA, 1999. Morgan Kaufmann Publishers.
11. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.
12. M. N. Rosenbluth and A. W. Rosenbluth. Monte Carlo calculation of the average extension of molecular chains. *Journal of Chemical Physics*, 23(256), 1955.
13. R. Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, 1981.
14. C. Yuan and M. J. Druzdzel. An importance sampling algorithm based on evidence pre-propagation. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 624–631, Morgan Kaufmann Publishers San Francisco, California, 2003.
15. C. Yuan and M. J. Druzdzel. Importance sampling algorithms for Bayesian networks: Principles and performance. *Mathematical and Computer Modelling*, 43:1189–1207, 2006.

# Adding Local Constraints to Bayesian Networks

Mark Crowley<sup>1</sup>, Brent Boerlage<sup>2</sup>, and David Poole<sup>3</sup>

<sup>1</sup> University of British Columbia, Vancouver, BC, Canada  
crowley@cs.ubc.ca

<sup>2</sup> Netica Division, Norsys Software Corp., Vancouver, Canada  
boerlage@norsys.com

<sup>3</sup> University of British Columbia, Vancouver, BC, Canada  
poole@cs.ubc.ca

**Abstract.** When using Bayesian networks, practitioners often express constraints among variables by conditioning a common child node to induce the desired distribution. For example, an ‘or’ constraint can be easily expressed by a node modelling a logical ‘or’ of its parents’ values being conditioned to true. This has the desired effect that at least one parent must be true. However, conditioning also alters the distributions of further ancestors in the network. In this paper we argue that these *side effects* are undesirable when constraints are added during model design. We describe a method called *shielding* to remove these side effects while remaining within the directed language of Bayesian networks. This method is then compared to chain graphs which allow undirected and directed edges and which model equivalent distributions. Thus, in addition to solving this common modelling problem, shielded Bayesian networks provide a novel method for implementing chain graphs with existing Bayesian network tools.

**Keywords:** Bayesian networks, constraints, mixed networks, chain graphs, graphical models, Bayesian modelling, complementary priors.

## 1 Introduction

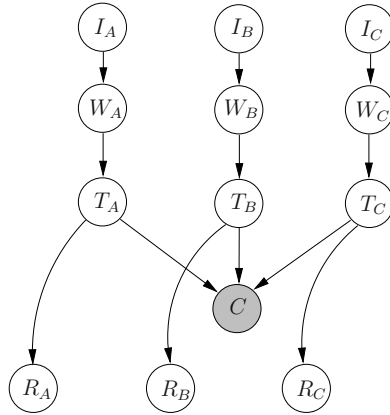
When using Bayesian networks it is often convenient to use conditioned nodes to enforce constraints across the network. Consider the following example:

*Example 1.* There are three professors, Alice, Bob and Cindy, at a university that needs at least one instructor for its AI course. For Alice we define four variables:  $I_A$ , modelling our belief that she is interested in AI; this influences  $W_A$ , our belief they she wants to teach the course; which influences  $T_A$ , our belief that she will actually end up teaching the course; which influences  $R_A$ , our belief that she completes her current research project on time. Variables are defined analogously for Bob and Cindy. The joint distribution of the variables  $T_A, T_B, T_C$  is consistent with the constraint that at least one professor must teach the course.

A natural way to represent this distribution is to add a node,  $C$ , to the network that models an ‘or’ of its parents and is conditioned to true. Figure 1 shows the Bayesian network that results. This enforces<sup>1</sup> the desired constraint onto the variables  $T_A, T_B, T_C$

---

<sup>1</sup> There are other ways to achieve this type of distribution without conditioning but it requires many new variables to be added and is difficult to maintain, see [1] for more details.



**Fig. 1.** Topic interests and teaching desires of three professors.  $C$  is an ‘or’ node stating that someone must teach the course.

which we call the *affected nodes*. This is similar to what [2] calls adding constraints using an auxiliary network. That paper models constraints by merging constraint network formalisms into Bayesian networks.

The desired distribution is one where the CPDs of all nodes express the probabilities *given* the presence of the constraint on the affected nodes. Thus, if  $p(I_A = \text{true}) = .7$  then we want  $p(I_A = \text{true}|C) = .7$ . But in a standard BN this will not be the case.  $I_A$  will be influenced by  $C$ , we call this influence a *side effect* of  $C$ . The reason we don’t want side effects is that they arise from treating  $C$  as evidence and  $p(T_A|W_A)$  as a simple conditional distribution. In fact, for this model, neither of these is the case.  $C$  is merely a convenient way to express a constraint, it does not constitute evidence, and thus its value should not be used freely for inference amongst its ancestors. But the constraint must be satisfied amongst its parents and the distribution on  $T_A$  is defined given the constraint.  $p(T_A|W_A)$  actually defines the probability distribution of  $T_A$  given that someone else has already been assigned to teach the course.

Thus, the constraint should have no influence on  $I_A$ . Our beliefs about Alice’s interest in AI are tied to the likely teaching assignments but are decoupled from the constraints on those teaching assignments. Altering or observing Cindy’s interest in AI should have no impact on Alice’s interest. On the other hand,  $R_A$  is influenced by the constraint. Research productivity is directly influenced by teaching assignments and so anything that impacts this must be taken into account when determining the likelihood of  $R_A$ .

Here we present a method to eliminate these side effects while maintaining a fully directed model and using existing Bayesian network tools. We will compare this method to chain graphs [3] which represent the same set of distributions by defining away the possibility of side effects. This paper has the dual goal of explaining how to solve a practical modelling problem with existing tools, as well as giving a new interpretation of chain graphs in terms of fully directed models.

## 2 Bayesian Networks

A Bayesian network (BN) [4] is a directed acyclic graph that represents the interdependence amongst a set of random variables. Suppose the variables are  $V_1, \dots, V_n$ . The Bayesian network represents the following factorization for the joint probability of a set of nodes in a Bayesian network:

$$p(V_1, \dots, V_n) = \prod_{i=1}^n p(V_i | pa(V_i)) \quad (1)$$

where  $pa(V_i)$  are the parent nodes on which  $V_i$  is dependant, if any.

### 2.1 Types of Conditioning

*Conditioning* refers to the general technique of setting a variable to a particular value within a BN. There are, at least, three types of conditioning. The most common type is simply recording an observation about the state of a variable or *observation conditioning*. The value here represents new information that rules out possible worlds that are incompatible with the observation. The remaining worlds are then renormalized to sum to 1. An observation can influence all of its ancestors and their descendants. If a variable is set by the user arbitrarily we call this *intervention conditioning* [5]. In this case the variable is set to some value by a mechanism outside of the model and so is not indicative of the variable's distribution. Thus the intervention cannot be used for inference about influences on the variables. Decision variables are of this type. An intervention should be cut off from influencing its ancestors but still influences its descendants.

A third type of conditioning, *constraint conditioning*, is the type being addressed in this paper. A node's value is set as part of the model definition in order to induce a particular distribution amongst its parent nodes. Other ancestors should not be affected just as they are not affected by the initial distributions of any other descendants. Influence on ancestors is cut off, just as in intervention, but in this case one level of nodes are allowed to be influenced. All of the descendants of these parents will then be influenced in the usual way. In this paper the constraint conditioned nodes such as  $C$  will be called *c-nodes*. The nodes in the constraint will (the parents) are the affected nodes or *e-nodes*. Nodes that are parents of affected nodes but are not themselves affected are known as *shielded nodes* or *s-nodes*.

We believe this is an important modelling problem for BNs. Bayesian networks are used widely every day for a broad range of purposes. We know from discussions with practitioners and experience that constraint conditioning is often used in practice. This is done as a natural extension of BN modelling and the full ramifications of side effects on the model may not always be realized. It is important for this issue to be widely discussed and possible solutions or alternative modelling techniques provided.

## 3 Removing Side Effects

Our goal now is to construct a BN in such a way that after inference is carried out the constraint conditioned nodes will have the desired influence and no more. We call this

method *shielding*. The chief insight is that we can add more conditioned nodes to cancel out the side effects. So, after adding a node  $\hat{C}$ , which we will define momentarily, we want the following to be true:

$$\begin{aligned} p(W_A|I_A, c, \hat{c}) &= p(W_A|I_A) & p(W_B|I_B, c, \hat{c}) &= p(W_B|I_B) \\ p(W_C|I_C, c, \hat{c}) &= p(W_C|I_C) & p(W_A, W_B, W_C|c, \hat{c}) &= p(W_A, W_B, W_C) \end{aligned} \quad (2)$$

where  $c$  indicates that  $C = c$ .

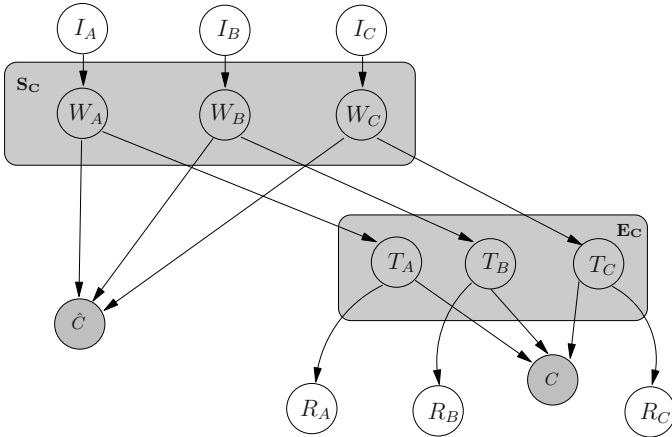
### 3.1 Antifactors

To define  $\hat{C}$  we need to think about how inference is carried out. A *factor* is the result of summing out some variables in a network during inference using a technique such as variable elimination [6][7]. In our example, if the affected nodes are summed out a factor is obtained,  $f_{T_{ABC}}(W_A, W_B, W_C)$ , representing the combined effect of the constraint on the  $W_A, W_B$  and  $W_C$  nodes. To cancel this we create an *antifactor* node,  $\hat{C}$ , with these nodes as parents, see Figure 2. The distribution of  $\hat{C}$  is defined by inverting the factor for the affected nodes as follows:

$$p(\hat{c}|W_A, W_B, W_C) = \frac{1}{Z} \frac{1}{f_{T_{ABC}}(W_A, W_B, W_C)} \quad (3)$$

where  $f_{T_{ABC}}(W_A, W_B, W_C) = \sum_{\mathbf{T}} p(c|T_A, T_B, T_C)p(T_A|W_A)p(T_B|W_B)p(T_C|W_C)$

The constant,  $Z$ , ensures that all values are in the range  $[0,1]$ . During inference this will cause the distributions of  $\hat{C}$  and the nodes  $C, T_A, T_B$  and  $T_C$  to exactly cancel each other making the distribution consistent with (2).



**Fig. 2.** An antifactor  $\hat{C}$  shields the influence of the  $c$ -node  $C$

### 3.2 General Antifactors

We now define the problem more generally.

**Definition 1.** A shielded Bayesian network (SBN) as satisfying the following requirement:

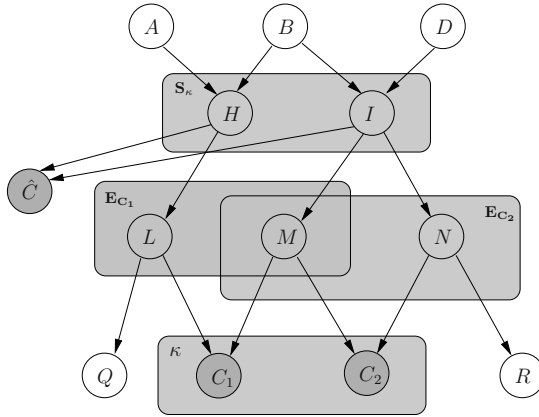
$$p(\mathbf{S}_C | c, \hat{c}) = p(\mathbf{S}_C) \quad (4)$$

Where  $C$  is a  $c$ -node and  $\hat{C}$  is a conditioned node added to the network with a distribution constructed to satisfy (4). The set  $\mathbf{E}_C = pa(\mathbf{E}_C)$  contains the affected nodes and  $\mathbf{S}_C = pa(pa(C)) - \mathbf{E}_C$  contains the shielded nodes. We assume there is no node in that is both an ancestor and a descendant of nodes in  $\mathbf{E}_C$ .

This can be satisfied by creating an antifactor node,  $\hat{C}$ , with parents  $\mathbf{S}_C$  such that

$$p(\hat{c} | \mathbf{S}_C) = \frac{1}{Z} \frac{1}{f_{\mathbf{E}_C}(\mathbf{S}_C)}$$

A more general case is shown in Figure 3. Here  $C_1$  and  $C_2$  are connected in a component because they share parents. Let  $\kappa$  be the minimum set of pairwise, disjoint components. The set  $\mathbf{S}_\kappa$  then denotes all the nodes to be shielded from every  $c$ -node in  $\kappa$ . An antifactor,  $\hat{C}$ , is defined with parents  $\mathbf{S}_\kappa$ . Its distribution is computed by summing out all nodes in  $\mathbf{E}_\kappa = E_{C_1} \cup E_{C_2}$ .



**Fig. 3.** The nodes  $L, M, N$  are constrained by two  $c$ -nodes with  $\kappa = \{C_1, C_2\}$ . The antifactor,  $\hat{C}$ , cancels out the effect on the  $\mathbf{S}_\kappa$  nodes.

**Definition 2.** With  $\kappa$  as a, possibly singleton, set of connected  $c$ -nodes and  $\hat{C}$  as its corresponding antifactor node, the following is the general definition of shielding:

$$p(\mathbf{S}_\kappa | \kappa, \hat{c}) = p(\mathbf{S}_\kappa) \quad (5)$$

Note that the antifactor *always exists* except in the case where the factor  $f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)$  contains a zero term. This occurs when the distribution of the affected network assigns a probability of zero to some instance of  $\mathbf{S}_\kappa$  after all the affected nodes have been summed out.

## 4 Antinetworks

The major drawback of using antifactors is complexity.  $\hat{C}$  connects all of the nodes in  $\mathbf{S}_\kappa$  creating a large new clique in the network. We could improve complexity by creating a conditional structure to reduce the number of parents of the antifactor.

An *antinetwork* is a set of nodes that mimic the structure of the original  $c$ -node and its parents. The distributions of the copied  $\hat{C}$  and  $\mathbf{E}_{\hat{\kappa}}$  nodes are computed so that summing out  $\mathbf{E}_{\hat{\kappa}}$  will yield  $\frac{1}{f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)}$ . Figure 4 shows the antinetwork for our example.

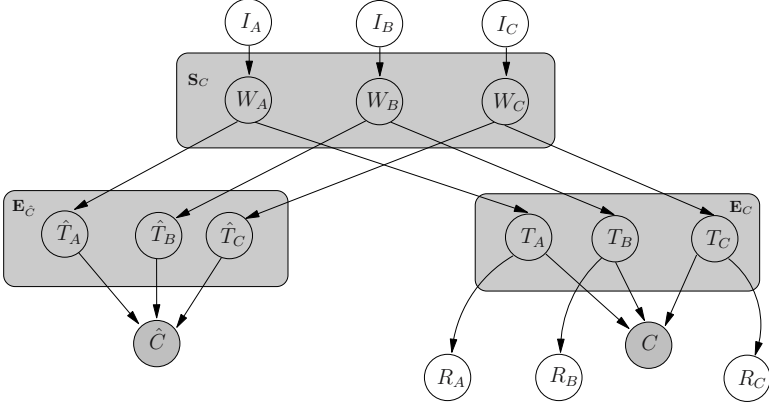


Fig. 4. An antinetwork shields the influence of  $c$ -node  $C$

### 4.1 Existence of a Solution

Unlike the antifactor solution it is not certain that a proper set of parameters for the antinetwork always exists although we have found them in many cases. Here we discuss some general properties of solutions.

The parameters to be solved for the antinetwork conform to the following system of equations. For simplicity, the case with binary nodes is shown here.

$$\begin{aligned}
 \text{Let } \pi &= f_{\mathbf{E}_{\hat{\kappa}}}(\mathbf{S}_\kappa) = \frac{1}{f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)} \\
 \pi &= \sum_{x \in \mathbf{E}_{\hat{\kappa}}} \prod_{\hat{c} \in \hat{\kappa}} p(\hat{c} | \mathbf{E}_{\hat{c}}) \prod_{\hat{e} \in x} p(\hat{e} | \mathbf{S}_{\hat{c}}) \\
 0 &= \sum_{x \in \mathbf{E}_{\hat{\kappa}}} \prod_{\hat{c} \in \hat{\kappa}} \gamma_{\hat{c}} \prod_{\hat{e} \in x} (\psi_{\hat{e}})^{(\hat{e}=\mathbf{t})} (1 - \psi_{\hat{e}})^{(\hat{e}=\mathbf{f})} - \pi = g_s(X)
 \end{aligned} \tag{6}$$

Where  $x$  captures one assignment to all the nodes in  $\mathbf{E}_{\hat{\kappa}}$  and the indicator exponent ( $\hat{e} = \mathbf{t}$ ) is simply 1 or 0. Note that this represents one equation for each instance  $s \in \mathbf{S}_\kappa$ . We will refer to this system as  $g_s(X)$  for  $X = \{\gamma_{\hat{c}}, \psi_{\hat{e}}\}$  for all  $\hat{c} \in \kappa$  and  $\hat{e} \in \mathbf{E}_{\hat{c}}$ . When  $X$  is found such that  $g_s(X) = 0$  then the antinetwork satisfies the shielding requirement.



**Solution Bounds.** When all the parameters are set to zero, denoted  $X_0$ , and one,  $X_1$ , the system yields:

$$g_s(X_0) = -\pi \quad g_s(X_1) = 1 - \pi$$

Since  $\pi$  is normalized to be a probability we have

$$g_s(X_0) \leq 0 \leq g_s(X_1) \quad \text{for all } s \in \mathbf{S}_\kappa$$

Since  $g_s(X)$  is a continuous function for each  $s \in \mathbf{S}_\kappa$  we know there is a solution  $X_s$  such that  $g_s(X_s) = 0$ . Unfortunately, we have not yet been able to show that there is always a simultaneous solution,  $X_*$ , to these equations such that  $g_s(X_*) = 0$  for all  $s \in \mathbf{S}_\kappa$ . The solution  $X_*$  is easy to identify when found as all the functions will be zero. When  $X \neq X_*$ , the solution can be used as an approximation to the correctly shielded distribution.

## 4.2 Finding a Solution

The antinetwork parameters can be solved by framing them as a *constrained optimization problem*. The objective function is a linear combination of the  $g_s(X)$  functions. The same functions are also used to define nonlinear, inequality constraints of the form  $g_s(X) \geq 0$ . Optimization is then begun at some known positive point, such as  $X_1$  and minimized until all  $g_s(X) = 0$ . See [1] for more details.

## 4.3 Solution Example

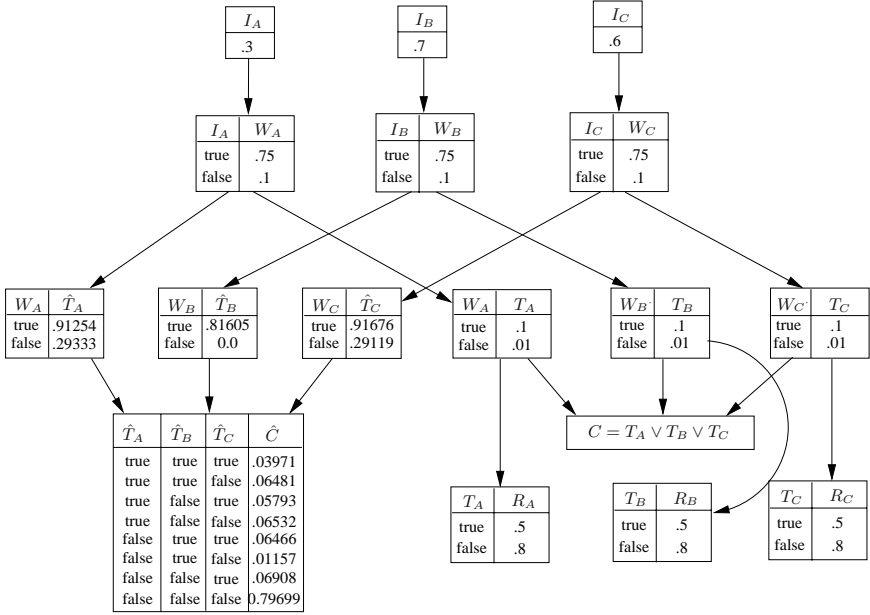
The solved CPDs for the antinetwork nodes, for Example 1, are shown in figure 5. The posteriors of the shielded nodes,  $W_A$ , and their ancestors,  $I_A$ , are correctly uninfluenced by the existence of the constraint. In particular, the ancestors maintain their prior distributions:

$$p(I_A = \mathbf{t} | c, \hat{c}) = .3 \quad p(I_B = \mathbf{t} | c, \hat{c}) = .7 \quad p(I_C = \mathbf{t} | c, \hat{c}) = .6$$

For networks where the affected sets overlap this method does not always find an exact solution. Our results approach the correct distribution but do not find an exact match. This indicates a solution may exist and that improved search techniques could yield a better approximation or an exact solution. When an exact solution is required an antifactor can always be used to shield the given  $c$ -nodes instead. Furthermore, antinetworks and antifactors can be used in the same network.

## 5 Undirected Models and Chain Graphs

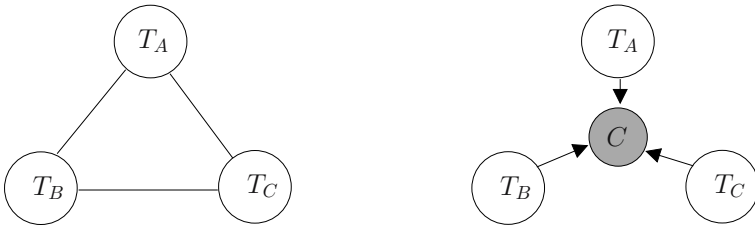
Another way to think about constraint conditioning is through undirected models. A Markov Random Field (MRF) [4] can easily be expressed as a BN by replacing all cliques potentials,  $\phi$ , with conditioned nodes. A simple construction makes this clear, see Figure 6:



**Fig. 5.** Computed CPDs for antinetwork found using nonlinear constrained optimization

- For each clique  $\mathcal{C}_i$  in the MRF, remove all links between nodes and replace with a directed link from each node in  $\mathcal{C}_i$  to a new binary node  $C_i$ .
- Assign the CPD of  $C_i$  such that  $p(C_i = \text{true} | \mathcal{C}_i) = \phi_i(\mathcal{C}_i)$
- Condition all of these added  $C_i$  nodes to be true.

These two representations model equivalent distributions. Note that under this construction the conditioned nodes will never have grandparents so shielding will not be needed. To model the full range of distributions we are interested in we need a combination of directed and undirected relationships.



**Fig. 6.** MRF to BN construction: a) A Markov Random Field. b) This MRF as a Bayes net with conditioned nodes replacing clique potentials.

## 5.1 Chain Graphs

A *chain graph* (CG) [3] is a graphical model that can have directed or undirected edges between its nodes. A *chain component*,  $\tau \in \mathcal{T}$ , is any set of nodes forming a connected component using undirected edges. Nodes in the directed portions of the network form their own chain components of size one. A CG can be seen as a directed, acyclic graph of chain components. The graph is acyclic in that there are no *partially directed cycles*. This is a cycle containing some directed edges, all pointing in the same direction around the cycle. Our example can be represented as a CG with the affected nodes represented as in figure 6(a) and other nodes connected as within the original BN.

The joint density of a CG is given by the following factorization [8] where the values of a set of variables  $\mathcal{V}$  is given by  $x_{\mathcal{V}}$ . Here  $\mathcal{A}(\tau)$  are all the fully connected sets of nodes from within  $\tau \cup pa(\tau)$ . Each of these has a clique potential  $\phi_A(x_A)$  across the nodes in the fully connected set. The  $Z$  term normalizes the density by summing across the values of all the nodes within the current chain component:

$$p(x_{\mathcal{V}}) = \prod_{\tau \in \mathcal{T}} p(x_{\tau} | x_{pa(\tau)}) \quad (7a)$$

$$p(x_{\tau} | x_{pa(\tau)}) = \frac{1}{Z(x_{\tau})} \prod_{A \in \mathcal{A}(\tau)} \phi_A(x_A) \quad (7b)$$

$$Z(x_{\tau}) = \sum_{x_{\tau}} \prod_{A \in \mathcal{A}(\tau)} \phi_A(x_A) \quad (7c)$$

Consider computing  $p(W_A)$ . It is clear that the distribution of the  $T_A, T_B, T_C$  chain component will play no part. This is because the nodes in the chain will be summed out in equation (7b) which will lead the  $\phi$  and  $Z$  terms to cancel exactly. In fact, lacking any observations, the variables  $W_A, W_B, W_C$  are independent of each other. CGs thus already express the kind of distribution we are concerned with where a joint constraint can exist amongst a set of variables without their ancestors being affected by the existence of that constraint. Note that in the presence of observations of a node in  $T_A, T_B$  or  $T_C$  this independence would no longer hold as this is new information that is relevant to all nodes.

## 5.2 Equivalence of SBNs and CGs

We will show the equivalence of shielded Bayesian networks with chain graphs by mapping each portion of SBNs to the factorization of CGs from equation (7).

1. The assumption in definition 1 is equivalent to the restriction against partially directed cycles in CGs.
2. Each chain component,  $\tau$ , in a CG has a corresponding set  $\kappa$  of *c-nodes* in an SBN.
3. Each potential function in the CG corresponds to the distribution of a *c-node* and the set of its affected nodes in the SBN

$$\phi_A(x_A) \propto p(c | \mathbf{E}_C) p(\mathbf{E}_C | \mathbf{S}_C).$$

4. The  $Z$  term is equivalent to marginalizing out the affected nodes. So for antifactors:

$$\frac{1}{Z(x_\tau)} \propto p(\hat{c}|\mathbf{S}_\kappa) = \frac{1}{f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)}. \quad (8)$$

And similarly for antinetworks:

$$\frac{1}{Z(x_\tau)} \propto p(\hat{c}|\mathbf{E}_\kappa) \prod_{\mathbf{E}_\kappa} p(\mathbf{E}_\kappa|\mathbf{S}_\kappa) = \frac{1}{f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)} \quad (9)$$

Note that if  $\kappa$  contains more than one  $c$ -node then they must be dealt with simultaneously.

With these mappings in place, the joint distributions in either model comes from a calculation that is equivalent up to a constant factor. This equivalence shows us that both models can be used to represent the same distribution.

The complexity of inference in graphical models is exponential in the size of the largest cliques in the network. We use clique in the same sense as in Junction trees [9], which are often used to perform inference in graphical models. Using either SBNs with antifactors or CGs this will be dominated by the size of the set  $\mathbf{S}_\kappa$ . This is because an antifactor has all of the nodes in  $\mathbf{S}_\kappa$  as its parents and so creates a clique of that size. Likewise, the potential function of a CG,  $\phi_A(x_A)$ , is defined over a *moralized graph* [10] where all the parents of nodes in a chain component are connected together. As we will show in the next section, antinetwork can avoid this blowup at least for some classes of networks.

## 6 Complexity Comparison

Consider the case where each  $e$ -node has  $m$  parents, none of which are shared with other  $e$ -nodes and all nodes have a domain of size  $D$ . This is the type of distribution described in Example 1. In all networks of this type tried an antinetwork solution has always been found.

The complexity for CGs is then exponential in the size of the clique  $A$  which is :

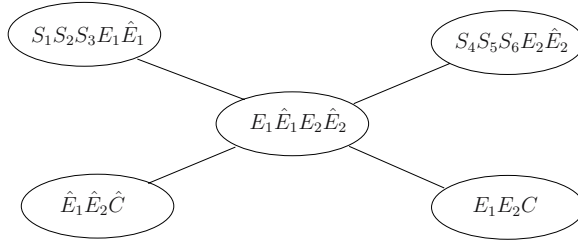
$$\begin{aligned} CG &= D^{|E_C|+|S_C|} \\ &= D^{|E_C|+m|E_C|} \quad \text{since } |S_C| = m \times |E_C| \\ &= D^{|E_C|(m+1)} \end{aligned} \quad (10)$$

For SBNs with antifactors, all of the  $s$ -nodes are combined into one clique. To maintain the triangulation property for junction trees each  $e$ -node is joined to all  $s$ -nodes. This leads to a slightly higher complexity than for CGs although the dominant term is the same as the CG complexity.

$$SBN_{\text{antifactor}} = D^{|E_C|+|S_C|} + D^{|S_C|} + D^{|E_C|} \quad (11)$$

An example of the junction tree for the third model, using antinetworks, is shown in figure 7. Conditional independence in the antinetwork reduces the complexity to:

$$SBN_{\text{antinetwork}} = D^{2|E_C|} + |E_C|D^{2+m} + 2D^{|E_C|} \quad (12)$$



**Fig. 7.** Junction tree for antinetwork model with  $|E_C| = 2$  and  $m = 3$

Thus, for this set of models we find that

$$SBN_{\text{antinetwork}} < CG \quad \text{when both } |E_C| \geq 2 \text{ and } m \geq 2.$$

So in general, as the number of parents of each *e-node* goes up, SBNs increase in complexity more slowly than CGs if the connectivity between ancestors of each *e-node* is low. When this is not the case, the antifactor method still provides a solution that has the same dominant term as the CG although it will have additional, smaller cliques as well.

## 7 Conclusion

In this paper we have formalized a common informal technique for adding constraints to BNs and pointed out serious side effects that may not be desired. The modeler faced with these unwanted side effects has several choices. They could re-evaluate their modelling assumptions, attempt to represent the constraint in other ways or use chain graphs instead. Modelers now have another option which is to use one of the shielding methods proposed here. The first method, antifactors, is universal and simple to apply but may be costly during inference. The second method, antinetworks, is more efficient for inference and while the empirical existence of solutions is promising there are no guarantees as of yet. The distributions modelled by these networks are equivalent to those of chain graphs. We have shown that at least for some classes of distribution, antinetworks are a more efficient representation than chain graphs. Further questions remain such as: Are there antinetwork solutions for wider classes of BNs? Are there any distributions that have compact antifactor solutions that would combine the advantage of both shielding methods? Can antifactors or antinetworks take advantage of context specific independence to reduce complexity?

There are strong similarities between our methods and complementary priors in [11] which offer intriguing lines of further research into learning. That work computes complementary priors quickly and efficiently to cancel out interdependence between network layers, it would be interesting to see if this can be applied to our modelling task. We believe the modeller's toolkit should include a variety of methods that allow flexibility to model any distribution needed. The techniques described here can be a very useful part of that toolbox when directed models and constraints are needed. Chain graphs are also available for these tasks but SBNs would be very natural to many modellers familiar with BNs. They require no extra tools beyond standard BN software to

be used and it would be straightforward to implement precompilers to automatically add antifactors or antinetworks to BNs. This solution contributes to Bayesian network modelling as well as adding insight into the relationships between all of these common modelling languages.

## References

1. Crowley, M.: Shielding against conditioning side effects in graphical models. Master's thesis, University of British Columbia, Canada (October 2005)
2. Dechter, R., Mateescu, R.: Mixtures of deterministic-probabilistic networks and their and/or search space. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-04), Arlington, Virginia, AUAI Press (2004) 120–129
3. Lauritzen, S.L., Wermuth, N.: Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics* **17** (1989) 31–57
4. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
5. Pearl, J.: Graphical models, causality and intervention. *Statistical Science* **8** (1993) 266–269
6. Zhang, N., Poole, D.: Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research* **5** (1996) 301–328
7. Dechter, R.: Bucket elimination : A unifying framework for probabilistic inference. In Horvitz, E., Jensen, F., eds.: Proceeding of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96). (1996) 211–219
8. Lauritzen, S.L., Richardson, T.S.: Chain graph models and their causal interpretations. *Journal of the Royal Statistical Society* **64**(3) (2002) 321–361
9. Jensen, F.V.: Junction trees and decomposable hypergraphs. Technical report, Judex Data-systemer, Aalborg, Denmark. (1988)
10. Lauritzen, S.L.: Graphical Models. Oxford: Clarendon (1996)
11. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. *Neural Comp.* **18**(7) (July 2006) 1527–1554

# On the Use of Possibilistic Bases for Local Computations in Product-Based Possibilistic Networks

Salem Benferhat and Salma Smaoui

CRIL, Université d'Artois, Faculté Jean Perrin,  
Rue Jean Souvraz, SP18, 62300 Lens, France  
{benferhat,smaoui}@cril.univ-artois.fr

**Abstract.** Product-based possibilistic networks allow an efficient representation of possibility distributions. However, when the graph is multiply connected, the propagation may be unfeasible because of the high space complexity problem. In this paper, we propose a new inference approach on product-based possibilistic networks based on compact representations of possibility distributions, which are possibilistic knowledge bases.

**Keywords:** Possibilistic networks, possibilistic logic, inference.

## 1 Introduction

Probabilistic and possibilistic networks are important graphical tools for representing and reasoning under uncertain pieces of information. In possibility theory, there are two kinds of possibilistic networks: min-based possibilistic networks and product-based possibilistic networks [1]. These two kinds of possibilistic networks only differ on the definition of possibilistic conditioning. Existing works for handling possibilistic inference through graphical models are mostly a direct adaptation of probabilistic approaches. In particular, for multiply-connected possibilistic networks, a graphical transformation from an initial possibilistic network to a junction tree (a tree of cliques) is achieved. This procedure allows to deal with many practical problems, however it fails when the number of variables in cliques are large, since it may be impossible to assign possibility distributions to cliques.

This paper proposes an alternative implementation of product-based possibilistic networks. We follow the same ideas that have recently been proposed for min-based possibilistic networks [2,3]. More precisely, we propose to use on cliques a compact representation of possibility distributions using possibilistic knowledge bases. In fact, properties of the possibilistic knowledge bases allow a more implicit representation of beliefs. This idea of combining logical-based representation and graphical have been previously considered by several authors [4,5,6,7]. In particular, Moral [6] uses local propagation algorithm for the deduction process in classical propositional logic. Our approach can be viewed as

an extension of their works, for non-idempotent operators, where propositional formulas are associated with necessity degrees.

The rest of this paper is organized as follows: first, we give a brief background on possibility theory and propagation algorithm for standard product-based possibilistic networks (section 2). Then, we present our new representation of the product-based possibilistic networks (Section 3). Section 4 details the different steps of the logic-based algorithm for product-based multiply connected graphs.

## 2 Possibility Theory

### 2.1 Notations

Let  $V = \{A_1, A_2, \dots, A_n\}$  be a set of variables.  $D_{A_i}$  denotes the finite domain associated with the variable  $A_i$ . For the sake of simplicity, and without loss of generality, variables considered here are assumed to be binary.  $a_i$  denotes any of the two instances of  $A_i$  and  $\neg a_i$  represents the other instance of  $A_i$ .  $\varphi, \psi, \dots$  denote propositional formulas obtained from  $V$  and logical connectors  $\wedge$  (conjunction),  $\vee$  (disjunction),  $\neg$  (propositional negation).  $\top$  and  $\perp$ , respectively, denote tautologies and contradictions.

$\Omega = \times_{A_i \in V} D_{A_i}$  represents the universe of discourse and  $\omega$ , an element of  $\Omega$ , is called an *interpretation*. It is either denoted by tuples  $(a_1, \dots, a_n)$  or by conjunctions  $(a_1 \wedge \dots \wedge a_n)$ , where  $a_i$ 's are respectively instance of  $A_i$ 's. In the following,  $\models$  denotes the propositional logic satisfaction.  $\omega \models \varphi$  means that  $\omega$  is a model of  $\varphi$ .

### 2.2 Possibility Distribution and Possibility Measure

One of the basic elements of possibility theory is the notion of possibility distribution  $\pi$  which is a mapping from  $\Omega$  to the interval  $[0, 1]$ . The degree  $\pi(\omega)$  represents the compatibility of  $\omega$  with available pieces of information. By convention,  $\pi(\omega) = 1$  means that  $\omega$  is totally possible, and  $\pi(\omega) = 0$  means that  $\omega$  is impossible. When  $\pi(\omega) > \pi(\omega')$ ,  $\omega$  is preferred to  $\omega'$  for being the real state of the world. A possibility distribution  $\pi$  is said to be **normalized** if there exists at least one interpretation which is consistent with available pieces of information. More formally,

$$\exists \omega \in \Omega, \pi(\omega) = 1$$

Uncertainty on an event  $\varphi \subset \Omega$  can be described by two dual measures: possibility measure  $\Pi$  and necessity measure  $N$ .

Considering a possibility distribution  $\pi$ , the possibility measure of a formula  $\varphi$  is as follows :

$$\Pi(\varphi) = \max\{\pi(\omega) : \omega \models \varphi\}$$

$\Pi(\varphi)$  represents the possibility degree that a model of  $\varphi$  exists in the real world. This measure evaluates the consistency level of  $\varphi$  with information encoded by  $\pi$ .

A necessity measure of a formula  $\varphi$  is defined as follows:

$$N(\varphi) = 1 - \Pi(\neg\varphi)$$



which corresponds to the certainty degree associated with  $\varphi$  from available pieces of information encoded by  $\pi$ .

### **Possibilistic conditioning**

Conditioning [8] is a crucial notion when dealing with independence relations. It consists of updating pieces of information encoded by  $\pi$  when an evidence (certain information)  $e$ , is observed. Let  $\varphi$  be the model set of  $e$ . Each  $\pi(\omega)$  is then replaced by  $\pi(\omega|\phi)$ .

In next sections, we will only consider the product-based conditioning, defined by:

$$\pi(\omega | \phi) = \begin{cases} \frac{\pi(\omega)}{\pi(\phi)} & \text{if } \omega \models \phi \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The product independence relation is obtained by using the product-based conditioning. Namely,

$$\Pi(x \wedge y|z) = \Pi(x|z) \cdot \Pi(y|z), \quad \forall x, y, z. \quad (2)$$

### **2.3 Possibilistic Logic**

A possibilistic knowledge base [9] is a finite weighted formula set:

$$\Sigma = \{(\varphi_i, \alpha_i) : i = 1, \dots, m\}$$

where  $\alpha_i$  is the lower-boundary of the necessity degree  $N(\varphi)$ . Namely,  $N(\varphi_i) \geq \alpha_i$ . Formulas with a necessity degree equal to 0 are not explicitly represented in the knowledge base. The more a formula is height-weighted the more it's certain. From each possibilistic knowledge base a unique possibility distribution is generated by associating each interpretation with a compatibility degree considering available information:

$\forall \omega \in \Omega,$

$$\pi_{\Sigma}(\omega) = \begin{cases} 1 & \text{if } \forall (\varphi_i, \alpha_i) \in \Sigma, \omega \models \varphi_i, \\ 1 - \max\{\alpha_i : \omega \not\models \varphi_i\} & \text{otherwise.} \end{cases} \quad (3)$$

The following definitions are useful for the rest of the paper:

**Definition 1.** Two possibilistic knowledge bases  $\Sigma_1$  and  $\Sigma_2$  are said to be equivalent if their associated possibility distributions are equal, namely :

$$\forall \omega \in \Omega, \quad \pi_{\Sigma_1}(\omega) = \pi_{\Sigma_2}(\omega)$$

Subsumption definition is as follows :

**Definition 2.** Let  $(\varphi, \alpha)$  a formula in  $\Sigma$ . Then  $(\varphi, \alpha)$  is said to be subsumed by  $\Sigma$  if  $\Sigma$  and  $\Sigma \setminus \{(\varphi, \alpha)\}$  are equivalent knowledge bases.

### **Coherence**

A possibilistic knowledge  $\Sigma$  is said to be consistent if its classical support, obtained by forgetting the weights, is classically consistent.

**Definition 3.** Let  $\Sigma$  be a possibilistic knowledge base. The inconsistency degree of  $\Sigma$ , denoted  $\text{Inc}(\Sigma)$ , is defined by :

$$\text{Inc}(\Sigma) = \max\{\alpha_i : \Sigma_{\geq \alpha_i} \models \perp\} \quad (4)$$

where  $\Sigma_{\geq \alpha_i}$  is a set of possibilistic formulas in  $\Sigma$  having a weight greater or equal to  $\alpha_i$ .

$\text{Inc}(\Sigma) = 0$  means that  $\Sigma$  is consistent.

Lang [10] proposed an algorithm to compute the inconsistency degree of  $\Sigma$  with a complexity equal to  $\log_2 n \text{ SAT}$  where  $n$  is the number of different valuations involved in  $\Sigma$ , and  $\text{SAT}$  is the propositional satisfiability test.

## 2.4 Standard Product-Based Possibilistic Networks

Possibilistic networks [1,11], denoted by  $\Pi G$ , are directed acyclic graphs (DAG). Nodes correspond to variables and edges encode relationships between variables. A node  $A_j$  is said to be a parent of  $A_i$  if there exists an edge from the node  $A_j$  to the node  $A_i$ . Parents of  $A_i$  are denoted by  $U_{A_i}$ .

Uncertainty is represented at each node by local conditional possibility distributions. More precisely, for each variable  $A$ :

If  $A$  is a root, namely  $U_A = \emptyset$ , then  $\max(\pi(a), \pi(\neg a)) = 1$ .

If  $A$  has parents, namely  $U_A \neq \emptyset$ , then  $\max(\pi(a|U_A), \pi(\neg a|U_A)) = 1$ , for each  $u_A \in D_{U_A}$ , where  $D_{U_A}$  is the cartesian product of domains of variables which are parents of  $A$ .

Possibilistic networks are also compact representations of possibility distributions. More precisely, joint possibility distributions associated with possibilistic network are computed using a so-called possibilistic chain rule similar to the probabilistic one, namely :

$$\pi_{\Pi G}(a_1, \dots, a_n) = \prod_{i=1..n} \Pi(a_i | u_{A_i}), \quad (5)$$

where  $a_i$  is an instance of  $A_i$  and  $u_{A_i} \subseteq \{a_1, \dots, a_n\}$  is an element of the cartesian product of domains associated with variables  $U_{A_i}$  which are parents of  $A_i$ .

**Example 1.** Figure 1 gives an example of possibilistic networks. Table 1 provides local conditional possibility distributions of each node given its parents.

Using possibilistic chain rule, the possibility degree of  $\pi(\neg ab \neg cd)$  is computed as follows:  $\pi(\neg ab \neg cd) = \pi(\neg a) \cdot \pi(b|\neg a) \cdot \pi(\neg c|\neg a) \cdot \pi(d|b \neg c) = 1 \cdot \frac{1}{4} \cdot 1 \cdot 1 = \frac{1}{4}$

## Inference algorithm in multiply-connected possibilistic networks

When DAGs are singly connected then the propagation algorithm is polynomial [11,1,12]. In this section, we only focus on multiply connected graphs.

A well-known propagation algorithm for multiply connected graphs proceeds to a transformation of the initial graph into a junction tree. The main steps of the junction tree construction are:

- Moralization of the initial DAG: Create an undirected graph from the initial graph and add links between parents of a common variable.

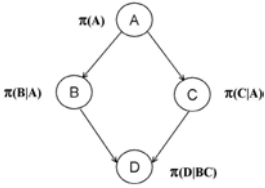


Fig. 1. Example of possibilistic network  $\Pi G$

Table 1. Local conditional possibility distributions associated with DAG of Figure 1

<table><tr><td><math>a</math></td><td><math>\frac{1}{4}</math></td></tr><tr><td><math>\neg a</math></td><td>1</td></tr></table>		$a$	$\frac{1}{4}$	$\neg a$	1	<table><tr><td><math>B A</math></td><td><math>a</math></td><td><math>\neg a</math></td></tr><tr><td><math>b</math></td><td><math>\frac{1}{4}</math></td><td><math>\frac{1}{4}</math></td></tr><tr><td><math>\neg b</math></td><td>1</td><td>1</td></tr></table>		$B A$	$a$	$\neg a$	$b$	$\frac{1}{4}$	$\frac{1}{4}$	$\neg b$	1	1									
$a$	$\frac{1}{4}$																								
$\neg a$	1																								
$B A$	$a$	$\neg a$																							
$b$	$\frac{1}{4}$	$\frac{1}{4}$																							
$\neg b$	1	1																							
<table><tr><td><math>C A</math></td><td><math>a</math></td><td><math>\neg a</math></td></tr><tr><td><math>c</math></td><td>1</td><td><math>\frac{1}{2}</math></td></tr><tr><td><math>\neg c</math></td><td><math>\frac{3}{4}</math></td><td>1</td></tr></table>		$C A$	$a$	$\neg a$	$c$	1	$\frac{1}{2}$	$\neg c$	$\frac{3}{4}$	1	<table><tr><td><math>D BC</math></td><td><math>bc</math></td><td><math>\neg bc</math></td><td>else</td></tr><tr><td><math>d</math></td><td>1</td><td><math>\frac{1}{4}</math></td><td>1</td></tr><tr><td><math>\neg d</math></td><td><math>\frac{1}{2}</math></td><td>1</td><td>1</td></tr></table>			$D BC$	$bc$	$\neg bc$	else	$d$	1	$\frac{1}{4}$	1	$\neg d$	$\frac{1}{2}$	1	1
$C A$	$a$	$\neg a$																							
$c$	1	$\frac{1}{2}$																							
$\neg c$	$\frac{3}{4}$	1																							
$D BC$	$bc$	$\neg bc$	else																						
$d$	1	$\frac{1}{4}$	1																						
$\neg d$	$\frac{1}{2}$	1	1																						

- Triangulation of the moral graph: Consists of adding edges to connect non-adjacent nodes in cycles of length four or greater.
- Building a junction tree from the triangulated moral graph: Consists of the junction tree construction by choosing the appropriate cliques and separators from the triangulated graph.

The main idea of these steps is to delete loops from the initial graph gathering some variables in a same node. The resulting graph is a tree, denoted  $AJ$ , where each node, called clique, is a set of variables. Common variables of two adjacent cliques are grouped into another type of node, called a separator.

Figure 2 gives an example of a junction tree associated with the DAG of figure 1 (there are two cliques  $\{ABC\}$  and  $\{BCD\}$  and one separator  $\{BC\}$  which is the intersection of the two cliques).



Fig. 2. Junction tree associated with graph  $\Pi G$  of figure 1

The propagation algorithm is then applied on this resulting structure. The idea is to require that adjacent cliques sharing common variables should have the same marginal distributions with respect to these common variables namely on their separator.

More precisely, the propagation algorithm consists of associating to each clique  $C_i$  a possibility distribution  $\pi_{C_i}$  which is a combination of initial local possibility distributions at the level of variables in the initial graph.

First, all possibility distributions are initialized to 1. Then,  $\forall A$  in the initial graph, we select a clique  $C_i$  containing  $\{A\} \cup U_A$  ( $U_A$ : parents of  $A$ ) and we update  $\pi_{C_i}$  as follows:

$$\pi_{C_i} \leftarrow \pi_{C_i} \cdot \pi(A|U_A)$$

Once the local possibility distributions  $\pi_{C_i}$  are initialized, the propagation process steps are realized through message passing between cliques until reaching stability:

$$\forall C_i, C_j \in A \mathcal{J} \quad \pi_{C_i}(\omega) = \pi_{C_j}(\omega)$$

Let  $M_{ij}(S_{ij})$  (resp.  $M_{ji}(S_{ij})$ ) be the message sent from  $C_i$  (resp.  $C_j$ ) to  $C_j$  (resp.  $C_i$ ). This algorithm is different from the one proposed by [11,1,12] since a clique  $C_j$  receiving a message does not update its local distribution but use the new received information by the message  $M_{ji}(S_{ij})$  to sent its own messages to other neighboring cliques. The local distributions are computed once all messages are sent. Then, each clique  $C_i$  receives from all its adjacent cliques  $C_k$  other than  $C_j$  a message  $M_{ki}(S_{ki})$  before sending its message to  $C_j$  given by:

$$M_{ij}(s_{ij}) = \max_{C_i \setminus S_{ij}} \pi_{C_i}(C_i) \prod_{k \neq j} M_{ki}(s_{ki}) \quad (6)$$

which is the combination between information received from neighbors and local distribution.

After sending all messages, the local distribution  $C_i$  is computed by combining local information by messages received from all its neighbors:

$$\pi_{C_i} = \pi_{C_i}(C_i) \prod_{k=1}^m M_{ki}(S_{ik}) \quad (7)$$

where  $m$  is the number of adjacent cliques of  $C_i$ .

When the junction tree is globally consistent,  $\pi(A)$  can be computed from any clique containing  $A$  by applying marginalization:

$$\pi(A) = \max_{C_i \setminus A} \pi_{C_i} \quad (8)$$

### 3 Possibilistic Logic-Based Networks

This section presents our new framework called possibilistic logic-based networks.

**Logic-based representation.** Before presenting the new representation, it remains to find the syntactical equivalent of the product operator. Let  $\pi_1$  and  $\pi_2$  be possibilistic distributions and  $\pi^* = \pi_1 \cdot \pi_2$  (the product of the two distributions). Let  $\Sigma_1$  and  $\Sigma_2$  be the possibilistic knowledge bases associated with  $\pi_1$  and  $\pi_2$  respectively. We need to find the knowledge base  $\Sigma^*$  associated with  $\pi^*$  [13]:

**Proposition 1.** *The resulting possibilistic knowledge bases associated with  $\pi^*$  is:*

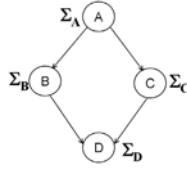
$$\Sigma_{(\Sigma_1, \Sigma_2)}^* = \Sigma_1 \cup \Sigma_2 \cup \{\varphi_i \vee \psi_j, \alpha_i + \beta_j - \alpha_i * \beta_j\}$$

where  $(\varphi_i, \alpha_i) \in \Sigma_1$  and  $(\psi_j, \beta_j) \in \Sigma_2$ .

The combination can be applied to several elements without taking care of the ordering in which elements are considered.

Our new representation framework is called a **logic-based network**, denoted  $LG$ , and is defined as follows:

- A *graphical component* which is represented by a DAG (like standard possibilistic networks) that allows to represent independence relationships.
- A *quantitative component* which encodes uncertainties. It associates to each node  $A$  a local knowledge base  $\Sigma_A$  (on  $A$  and its parents) instead of a conditional possibility distribution.



**Fig. 3.** Logic-based graph  $LG$  with local knowledge bases

Logic-based graphs are also compact representations of joint possibility distributions. A possibility distribution associated with a logic-based possibilistic network  $LG$  is defined by:

$$\forall \omega, \pi_{LG}(\omega) = \prod_{A \in V} \pi_{\Sigma_A}(\omega) \quad (9)$$

where  $\pi_{\Sigma_A}$  is the possibility distributions associated with  $\Sigma_A$  obtained using equation (3).

Any standard possibilistic network  $\Pi G$  can be equivalently represented by a logic-based network  $LG$ . Let  $A$  be a variable, and  $\pi(a_i|u_i)$  be a local possibility degree associated with  $A$ . Then the logic-based possibilistic network  $LG$  associated with  $\Pi G$  is obtained in the following way: for each  $A$ , define

$$\Sigma_A = \{(\neg a_i \vee \neg u_i, \alpha_i) : \alpha_i = 1 - \pi(a_i|u_i) \neq 0\}. \quad (10)$$

Then we have:

**Proposition 2.** *Let  $\Pi G$  be a standard possibilistic network. Let  $LG$  be a logic-based network having the same DAG and where  $\Sigma_{A_i}$ 's are obtained from the local possibility distributions  $\pi_{\Pi G}(A_i|U_i)$  in  $\Pi G$  using equation (10), then,*

$$\pi_{\Pi G}(\omega) = \pi_{LG}(\omega) \quad (11)$$

where  $\pi_{\Pi G}$  and  $\pi_{LG}$  are obtained by using (5) and (9).

This approach has an intuitive appeal. Namely, possibilistic knowledge bases allow a more implicit representation:

- possibilistic formulas having a weight equal to 0 are not represented in the possibilistic knowledge base,
- several formulas may be subsumed by other ones.

**Example 2.** *Let us build a logic-based possibilistic network  $LG$  from product-based possibilistic network  $\Pi G$  of example 1 by assigning a knowledge base to each node using 10. Uncertainty at the level of nodes  $A, B, C$  and  $D$  (binary variables) is represented by possibilistic knowledge bases  $\Sigma_A, \Sigma_B, \Sigma_C$  and  $\Sigma_D$  as follows:*

$$\Sigma_A = \{(\neg a, \frac{3}{4})\}; \Sigma_B = \{(\neg a \vee \neg b, \frac{3}{4}), (a \vee \neg b, \frac{3}{4})\}; \Sigma_C = \{(a \vee \neg c, \frac{1}{2}), (\neg a \vee c, \frac{1}{4})\}; \Sigma_D = \{(b \vee \neg c \vee \neg d, \frac{3}{4}), (\neg b \vee \neg c \vee d, \frac{1}{2})\}$$

*We can check that,  $\forall \omega, \pi_{\Pi G}(\omega) = \pi_{LG}(\omega)$ . For instance,  $\pi_{LG}(\neg ab \neg cd) = \pi_{\Sigma_A}(\neg ab \neg cd) \cdot \pi_{\Sigma_B}(\neg ab \neg cd) \cdot \pi_{\Sigma_C}(\neg ab \neg cd) \cdot \pi_{\Sigma_D}(\neg ab \neg cd) = 1 \cdot \frac{1}{4} \cdot 1 \cdot 1 = \frac{1}{4}$  which is the same result as in example 1.*

## 4 Possibilistic Inference in $LJT$

To apply the general propagation algorithm through junction trees, we need an equivalent structure using the new representation. This structure is called **logic-based junction tree** and is denoted  $LJT$ . It is characterized by:

- Graphical component: Graphical component is a junction tree built from the initial graph in a similar way as for standard networks.
- Numerical component: First, We associate an empty knowledge base  $\Sigma_{C_i}$  (resp.  $S_{ij}$ ) for each clique  $C_i$  (resp. separator  $S_{ij}$ ). Then, for all variable  $A$  in the initial graph, we select a clique  $C_i$  containing  $A \cup U_A$  ( $U_A$  parents of  $A$ ):

$$\Sigma_{C_i} \leftarrow \Sigma_{C_i} \cup \Sigma_A$$

where  $\Sigma_A$  is the local knowledge base in  $LG$  at the level of  $A$ .

Before introducing the propagation algorithm on the  $LJT$  structure, we need to present the notion of prioritized forgetting (see [14]) which allows to give the syntactic counterpart of the marginalization process.

### 4.1 Prioritized Forgetting

Lin and Reiter [14] proposed an approach allowing variable domain restriction in propositional knowledge bases (see [15,16] for details). Variable forgetting (also known as projection or marginalization) is defined as:

**Definition 4.** *Let  $K$  be a propositional knowledge base and  $X$  be a propositional variable set. The forgetting of  $X$  in  $K$ , noted  $\text{forget}(K, X)$ , is equivalent to a propositional formula that can be inductively defined as follows:*

- $\text{forget}(K, \emptyset) = K$ .
- $\text{forget}(K, \{x\}) = K_{x \leftarrow \perp} \vee K_{x \leftarrow \top}$ .

- $forget(K, X \cup \{x\}) = forget(forget(K, X), \{x\})$ .

where  $K_{x \leftarrow \perp}$  (resp.  $K_{x \leftarrow \top}$ ) refers to  $K$  in which we affect false (resp. true) value to each occurrence of  $x$  (instance of  $X$ ). By  $K_i \vee K_j$  we mean the set  $\{(\varphi_i \vee \psi_j) : \varphi_i \in K_i \text{ and } \psi_j \in K_j\}$ .

This approach is defined for classical propositional logic. We present an extension of this definition, called prioritized forgetting, which deals with possibilistic knowledge bases.

Let  $\Sigma_1$  and  $\Sigma_2$  be two possibilistic knowledge bases. The disjunction of two bases in possibilistic framework, denoted by  $\bigvee$ , is defined as follows:

$$\Sigma_1 \bigvee \Sigma_2 = \{(\varphi_i \vee \psi_j, \min(\alpha_i, \beta_j)) : (\varphi_i, \alpha_i) \in \Sigma_1 \text{ and } (\psi_j, \beta_j) \in \Sigma_2\}$$

Prioritized forgetting, denoted  $pforget$ , can then be defined as follows:

**Definition 5.** Let  $\Sigma$  be a possibilistic knowledge base and  $X$  be a variable set. The prioritized forgetting of  $X$  in  $\Sigma$ , denoted  $pforget(\Sigma, X)$ , is equivalent to a possibilistic formula defined as follows:

- $pforget(\Sigma, \emptyset) = \Sigma$ ,
- $pforget(\Sigma, \{x\}) = K_{x \leftarrow \perp} \bigvee K_{x \leftarrow \top}$
- $pforget(\Sigma, X \cup \{x\}) = pforget(pforget(\Sigma, X), \{x\})$ .

Prioritized forgetting allows to syntactically capture the base associated with marginal distributions. More precisely:

**Proposition 3.** Let  $\Sigma$  be a possibilistic knowledge base and  $\pi$  its associated distribution. Let  $X$  be a set of variables. Then the possibility distribution associated with  $pforget(\Sigma, X)$  is:

$$\pi_{pforget(\Sigma, X)} = \max_{V \setminus X} \pi_{\Sigma} \quad (12)$$

## 4.2 Propagation Algorithm

After defining the prioritized forgetting, we propose an alternative propagation algorithm in junction trees. In fact, our propagation algorithm can be viewed as the counterpart of the one proposed in Subsection 2.4 (for standard junction trees), where rather to use possibility distributions, we use possibilistic knowledge bases.

To illustrate this process, let  $LJT$  be the junction tree and let  $m$  be the number of cliques in  $LJT$ . Suppose that a clique  $C_i$  has  $q$  adjacent cliques  $\{B_1, \dots, B_q\}$ . Let  $C_{ij}$  and  $X_{ij}$  be the set of cliques and the variable set on the subtree containing  $C_i$  when dropping the link  $C_i - B_j$ .

Then,  $X = C_{ij} \cup C_{ji} = X_{ij} \cup X_{ji}$ .

The main steps of the logic-based junction tree algorithm are summarized in figure 5.

In next sections, we present each step details.

**Step 1-Initialization by handling evidence (if  $e \neq \emptyset$ ).** Handling evidence consists of computing for each variable  $A \in V$  the possibility degree  $\pi(A \wedge e)$  when  $e$  is the total evidence. To handle the evidence, we should extend the propagation

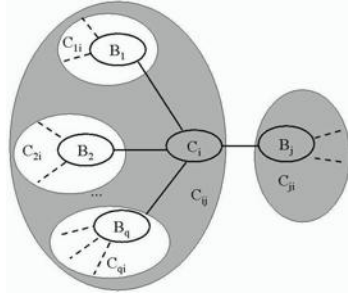


Fig. 4. Decomposition into disjoint sets

```

Begin
  - Apply step 1: Handling evidence (if  $e \neq \emptyset$ ),
  While (Junction tree is not consistent) do
    - Apply step 2: Computing messages (knowledge bases)  $M_{ij}$  and  $M_{ji}$ ,
       $\forall C_i, C_j$  adjacent cliques,
    done
    - Apply step 3: Computing the new local knowledge bases  $\Sigma_{C_i}$  for each clique  $C_i$ 
      using  $M_{ij}$  and  $M_{ji}$ ,
    - Apply step 4: Answering queries.
End

```

Fig. 5. Logic-based propagation algorithm

procedure by transforming the initial local knowledge bases  $\{\Sigma_{C_i} : C_i \in LJT\}$  at the level of the logic-based junction tree  $LJT$  as follows:

- For each observed variable  $A = a$ , select a clique  $C_i$  containing the variable  $A$ , and add the formula  $(a, 1)$  to the knowledge base associated with  $C_i$ .

In fact, adding the formula  $(a, 1)$  implies that,  $\forall \omega : \omega \models \neg a$ ;  $\pi(\omega) = 0$ .

After this step, messages are sent between cliques in order to guarantee the consistency condition.

**Step 2-Computing messages  $M_{ij}$ .** In order to be able to compute the messages  $M_{ij}$  and  $M_{ji}$  sent between the disjoint subtrees  $C_{ij}$  and  $C_{ji}$ , we first compute the knowledge base  $\Sigma_{S_{ij}}$  of a typical separator. Let us consider the following decomposition:

$$X \setminus S_{ij} = (X_{ij} \cup X_{ji}) \setminus S_{ij} = (X_{ij} \setminus S_{ij}) \cup (X_{ji} \setminus S_{ij}) = R_{ij} \cup R_{ji} \quad (13)$$

where  $R_{ij} = X_{ij} \setminus S_{ij}$ . It is trivial that  $R_{ij}$  and  $R_{ji}$  are disjoint subsets.

The possibilistic knowledge base  $\Sigma_{S_{ij}}$  associated with the separator  $S_{ij}$  is computed as follows:

$$\begin{aligned}
 \Sigma_{S_{ij}} &= pforget(\Sigma_{LJT}, X \setminus S_{ij}) \\
 &= pforget(\Sigma^*(\Sigma_{C_k} : k = 1, \dots, m), X \setminus S_{ij}) \\
 &= pforget(\Sigma^*(\Sigma_{C_k} : k = 1, \dots, m), R_{ij} \cup R_{ji}) \\
 &= \Sigma^*(pforget(\Sigma^*(\Sigma_{C_k} : C_k \in C_{ij}), R_{ij}), pforget(\Sigma^*(\Sigma_{C_k} : C_k \in C_{ji}), R_{ji})) \\
 &= \Sigma^*(M_{ij}, M_{ji})
 \end{aligned}$$



Let us consider this composition:

$$X_{ij} \setminus S_{ij} = (C_i \setminus S_{ij}) \cup \left( \bigcup_{k \neq j} X_{ki} \setminus S_{ij} \right) \quad (14)$$

Thus,  $\forall k, k \neq j$ , we obtain:

$$\begin{aligned} M_{ij} &= pforget(\Sigma^*(\Sigma_{C_s} : C_s \in C_{ij}), X_{ij} \setminus S_{ij}) \\ &= pforget(pforget(\Sigma^*(\Sigma_{C_s} : C_s \in C_{ij}), X_{ki} \setminus S_{ki}), X_{ij} \setminus S_{ij}) \\ &= pforget(\Sigma^*(\Sigma_{C_i}, \Sigma^*(\{pforget(\Sigma^*(\Sigma_{C_s} : C_s \in C_{ij}), X_{ki} \setminus S_{ki})\})), C_i \setminus S_{ij}) \\ &= pforget(\Sigma^*(\Sigma_{C_i}, \Sigma^*(M_{ki} : k \neq j)), C_i \setminus S_{ij}) \end{aligned}$$

since information at the level of  $C_i$  is the combination of the local knowledge and the knowledge coming from all neighboring cliques other than  $B_j$ .

Then, message sent from  $C_i$  to  $B_j$  is:

$$M_{ij} = pforget(\Sigma^*(\Sigma_{C_i}, \Sigma^*(M_{ki} : k \neq j)), C_i \setminus S_{ij})$$

The message sent from  $B_j$  to  $C_i$  is:

$$M_{ji} = pforget(\Sigma^*(\Sigma_{B_j}, \Sigma^*(M_{kj} : k \neq i)), B_j \setminus S_{ij})$$

The possibilistic knowledge base associated with the separator  $S_{ij}$  is simply the syntactical equivalent of the product  $\Sigma^*$  of these two messages. These messages contain all relevant information. Namely, they gather information from one side of the link and propagate it to the other side. The last equation shows that the message  $M_{ij}$  sent by the clique  $C_i$  to its adjacent clique  $B_j$  can be computed as soon as  $C_i$  receives all messages  $\{M_{ki} : k \neq j\}$ .

**Step 3-Computing local possibilistic knowledge bases  $\Sigma_{C_i}$ .** After receiving all messages  $\{M_{ki} : k = 1, \dots, q\}$  from all its neighboring cliques, the clique  $C_i$  computes its local knowledge base  $\Sigma_{C_i}$ . Let us consider the following decomposition:

$$X \setminus C_i = \left( \bigcup_{k=1}^q X_{ki} \right) \setminus C_i = \bigcup_{k=1}^q (X_{ki} \setminus C_i) = \bigcup_{k=1}^q R_{ki} \quad (15)$$

Then, the JPD of  $C_i$  is given as the following:

$$\begin{aligned} \Sigma_{C_i} &= pforget(\Sigma_{LJT}, X \setminus C_i) \\ &= pforget(\Sigma^*(\Sigma_{C_j} : j = 1, \dots, m), X \setminus C_i) \\ &= \Sigma^*(\Sigma_{C_i}, pforget(\Sigma^*(\Sigma_{C_j} : j \neq i), X \setminus C_i)) \\ &= \Sigma^*(\Sigma_{C_i}, pforget(\Sigma^*(\Sigma_{C_j} : j \neq i), R_{1i} \cup \dots \cup R_{qi})) \\ &= \Sigma^*(\Sigma_{C_i}, pforget(\Sigma^*(\Sigma_{C_k} : C_k \in C_{1i}), R_{1i}), \\ &\quad \dots, pforget(\Sigma^*(\Sigma_{C_k} : C_k \in C_{qi}), R_{qi})) \\ &= \Sigma^*(\Sigma_{C_i}, \Sigma^*(M_{ki} : k = 1, \dots, q)) \end{aligned}$$

This equation shows that the local knowledge base  $\Sigma_{C_i}$  can be computed as soon as the clique  $C_i$  receives all messages from its neighboring cliques.

After sending all messages and computing all local possibilistic knowledge bases, the junction tree is called "globally consistent". Formally,  $LJT$  is consistent if  $\forall i, j$ , we have:

$$\Sigma_{S_{ij}} = pforget(\Sigma_{C_i}, C_i \setminus S_{ij}) = pforget(\Sigma_{C_j}, C_j \setminus S_{ij}) \quad (16)$$

**Proposition 4.** *The joint possibility distribution  $\pi_{LJT}$  associated with the junction tree after sending all messages is equivalent to the initial distribution  $\pi_{LG}$  (associated with the initial graph):*

$$\pi_{LG} = \pi_{LJT} \quad (17)$$

**Example 3.** *The knowledge base  $\Sigma_{C_2}$  associated with the clique  $C_2$  after receiving  $\Sigma_{S_{12}}$  is:*

$\Sigma_{C_2} = \Sigma_{C_2} \cup \Sigma_{S_{12}} = \{(b \vee \neg c \vee \neg d, \frac{3}{4}), (\neg b \vee \neg c \vee d, \frac{1}{2}), (\neg b, \frac{3}{4}), (\neg c, \frac{1}{2})\}$  which is equivalent to  $\Sigma_{C_2} = \{(b \vee \neg c \vee \neg d, \frac{3}{4}), (\neg b, \frac{3}{4}), (\neg c, \frac{1}{2})\}$ .

*At the end of propagation process, we obtain the following local knowledge bases:*

- $\Sigma_{C_1} = \{(\neg a, \frac{3}{4}), (\neg b, \frac{3}{4}), (\neg c, \frac{1}{2})\}$ .
- $\Sigma_{C_2} = \{(\neg b, \frac{3}{4}), (\neg c, \frac{1}{2}), (b \vee \neg c \vee \neg d, \frac{3}{4})\}$ .

*It can be checked that  $LJT$  is consistent.*

**Step 4-Answering queries.** After the propagation process, the syntactical marginal with respect to  $V_i$  is computed at the level of clique  $C_i$ . Then, to compute  $\{\pi(A) : A \in V_i\}$ , it is enough to eliminate symbols from intermediate result on  $C_i$  which represents a computational simplification.

When the junction tree is consistent, computing  $\Pi(A)$  is done syntactically using possibilistic inference:

**Proposition 5.** *Let  $\Sigma$  be a possibilistic knowledge base. Let  $a$  be an instance of  $A$ . Then,*

$$\pi(a) = 1 - \text{Inc}(\Sigma \cup \{(a, 1)\})$$

*where  $\text{Inc}(\Sigma \cup \{(a, 1)\})$  is the inconsistency degree of  $\Sigma \cup \{(a, 1)\}$ . For computing the inconsistency degree  $\text{Inc}$  see [9].*

## 5 Conclusion

In this paper, we proposed a new inference algorithm for product-based possibilistic networks. Our framework called possibilistic logic-based networks allows to encode standard possibilistic networks. The algorithm consists of local computations using compact representations of possibility distributions which are possibilistic knowledge bases. Our approach is an interesting alternative implementation of well-known junction tree algorithms, since it is particularly appropriate when it is impossible to initialize local distributions on cliques (namely when cliques' sizes are large).

## Acknowledgment

This work is supported by the national project ANR Blanc MICRAC (Modèles Informatiques et Cognitifs du Raisonnement Causal).

## References

1. Fonck, P.: Réseaux d'inférence pour le raisonnement possibiliste. PhD thesis, Université de Liège, Faculté des Sciences (1994)
2. Benferhat, S., Smaoui, S.: Hybrid possibilistic networks. To appear in proceeding of the Twentieth National Conference on Artificial Intelligence (AAAI-05) (2005)
3. Benferhat, S., Smaoui, S.: Possibilistic networks with locally weighted knowledge bases. In: Fabio Gagliardi Cozman and Robert Nau and Teddy Seidenfeld, Proceedings of the Fourth International Symposium on Imprecise Probabilities and Their Applications (ISIPTA '05), Pittsburgh, USA, Brightdocs (2005) 41–50
4. Wilson, N., Mengin, J.: Embedding logics in the local computation framework. *Journal of Applied Non-Classical Logics* **11**(3-4) (2001) 239–267
5. Mengin, J., Wilson, N.: Logical deduction using the local computation framework. *Lecture Notes in Computer Science* (1999) 386–??
6. Hernandez, L., Moral, S.: Inference with idempotent valuations. In: Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97), San Francisco, CA, Morgan Kaufmann (1997) 229–237
7. Dubois, D., Prade, H.: Possibility theory: An approach to computerized, Processing of uncertainty. Plenum Press, New York (1988)
8. Hisdal, E.: Conditional possibilities independence and non interaction. *Fuzzy Sets and Systems* **1** (1978) 283–297
9. Dubois, D., Lang, J., Prade, H.: Possibilistic logic. In: Handbook on Logic in Artificial Intelligence and Logic Programming. Volume 3. Oxford University press (1994) 439–513
10. Lang, J.: Possibilistic logic: complexity and algorithms. *Handbook of Defeasible Reasoning and Uncertainty Management Systems* **5** (2000) 179–220
11. Borgelt, C., Gebhardt, J., Kruse, R.: Possibilistic graphical models. In: Proceedings of International School for the Synthesis of Expert Knowledge (ISSEK'98 ), Udine (Italy) (1998) 51–68
12. Gebhardt, J., Kruse, R.: Background and perspectives of possibilistic graphical models. In: Proceedings of European Conference of Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'1997), Bad Honnef (Germany) (1997) 108–121
13. Benferhat, S., Dubois, D., Prade, H.: Some syntactic approaches to the handling of inconsistent knowledge bases: A comparative study Part 2: The prioritized case. In Orlowska, E., ed.: *Logic at work*. Volume 24. Physica-Verlag, Heidelberg (1998) 473–511
14. Lin, F., Reiter, R.: Forget it! Proceeding of AAAI Fall Symposium on Relevance (1994) 154–159
15. Lang, J., Marquis, P.: Complexity results for independence and definability. Proceeding of the 6th International Conference on Knowledge Representation and Reasoning (KR'98) (1998) 356–367
16. Darwiche, A., Marquis, P.: Compiling propositional weighted bases. *Artif. Intell.* **157**(1-2) (2004) 81–113

# Reasoning with Conditional Preferences Across Attributes

Shaoju Chen<sup>1</sup>, Scott Buffett<sup>2</sup>, and Michael W. Fleming<sup>1</sup>

<sup>1</sup> University of New Brunswick, Fredericton, NB, E3B 5A3  
`{shaoju.chen,mwf}@unb.ca`

<sup>2</sup> National Research Council Canada, Fredericton, NB, E3B 9W4  
`Scott.Buffett@nrc.gc.ca`

**Abstract.** Before an autonomous agent can perform automated negotiation on behalf of a user in an electronic commerce transaction, the user's preferences over the set of outcomes must be learned as accurately as possible. This paper presents a structure, a Conditional Outcome Preference Network (COP-network), for modeling preferences directly elicited from a user. The COP-network then expands to indicate all preferences that can be inferred as a result. The network can be easily checked for consistency and redundancy, and can be used to determine quickly whether one outcome is preferred over another. An important feature of the COP-network is that conditional preferences, where a user's preference over outcomes depends on whether particular attribute values are included, can be modeled and inferred as well. If the agent also knows the user's utilities for some of the possible outcomes, then these can be considered in the COP-network as well. Three techniques for estimating utilities based on the specified preferences and utilities are described. One such technique, which works by first estimating utilities for long chains of outcomes for which preferences are known, is shown to be the most effective.

## 1 Introduction

The widespread use of the Internet today allows people to engage in more communication, interaction, and transactions online than ever before. Opportunities for automated negotiation between agents over the Internet are abundant, and the use of such technologies becomes more and more feasible as the speed of communication and processing increases. Buyers can negotiate with sellers over the price or other terms of a potential exchange. Web users can negotiate the terms of websites' policies for handling private information. Even the terms of use and configuration of web services can be negotiated. However, before negotiation can commence, an agent representing a user must know the user's utilities for potential agreements. Utility elicitation techniques can help the agent to learn some of the user's preferences, but it is typically infeasible to learn all of them due to the exponential number of outcomes. Therefore, an agent must gather as much information as possible from a few utility elicitation queries, and attempt to estimate or infer utilities over outcomes that cannot be elicited directly.

In the realm of privacy, as well as many other application areas, utilities for outcomes cannot typically be computed as an additive function of the user's utilities for individual aspects or attributes of those outcomes. This is due to the highly dependent nature of the attributes. Matters are complicated when a user specifies conditional preferences. For example, a user may not mind releasing information which identifies his place of employment, nor would he mind exposing his job title. However, he may have strong reservations when it comes to giving away both of these particular items of information, as it may personally identify him. So perhaps his utility for exposing his job title is dependent on whether his place of employment is also part of the final outcome.

One can envision other situations where this may be the case, such as in the stock market. Investors often prefer to create a balanced portfolio, where risks are hedged against each other and the chance of overall growth is maximized. Here, adding particular items to the portfolio will be more or less preferred, depending on which others are included. Another example is the case of determining options to be included in a new car. Perhaps air conditioning is important to a particular buyer, but becomes less important if a convertible roof is included.

Given this, one can see that it is quite complex to determine a global utility function that is consistent with all preferences that can be derived given the known interdependencies. In order to determine such utilities, a preference structure is needed. Boutilier et al [1,2] present a structure, known as a CP-network, for reasoning about conditional preferences over values within a single attribute. Consider a car example with attributes "Make" and "Colour". A user may specify preferences for "Make" such as "Pontiac is preferred over Volkswagen", or "Colour" such as "Black is preferred over silver". From this, the reasoning technique can infer that black Pontiacs are preferred over silver Volkswagens, all else equal. Additionally, conditional preferences can be used. For example, consider a buyer that only likes Pontiacs that were made after 2002. Then the preference for "make" is conditional on the outcome for "year".

In the privacy example, attributes of outcomes correspond to items of personal information to be exchanged. Each attribute can then take on one of two values: "included in the agreement" or "not included in the agreement". Under Boutilier's model, the user can only specify preferences such as for "e-mail address", "not included" is preferred over "included" (and likewise for "phone number"). In areas such as privacy where sets of items are being negotiated, a richer model is needed where preferences can be expressed across attributes, such as the value "included" is preferred for "phone number" over the value "included" for "e-mail address" (i.e., phone number is preferred over e-mail).

In this paper, we develop a preference structure that will indicate all preferences that can be derived, given the conditional and unconditional utilities elicited from the user. This structure is referred to as a Conditional Outcome Preference Network (COP-network or COPN). The COP-network is a directed graph where, given that the information elicited from the user is accurate, if an outcome  $o$  in the graph precedes another outcome  $o'$ , then the user's true utility

for  $o$  is higher than that for  $o'$ . This network is then used to estimate the user's utilities for all outcomes for which the utility has not been elicited.

The paper is organized as follows. In Section 2 we briefly review the CP-network, and in Section 3 we introduce our COP-network. In Section 4 we demonstrate how the COPN can be used to estimate utilities over outcomes, and give a simple example of how this would be applied. In Section 5 we discuss the results of our tests for accuracy of our technique, and in Sections 6 and 7 we offer our conclusions and discuss directions for future work.

## 2 Conditional Preference Networks (CP-Nets)

Boutilier et al. [1,2] explore a representation referred to as a conditional preference network (CP-network) for structuring user preferences. The representation is based on the dependence and conditional preferential independence between attributes. A CP-network over a set of  $n$  attributes is graphical, where a node is created for every attribute. For each attribute, the user must identify a set of parent attributes whose values can influence the user's preference over values for the attribute. Each node has an associated table describing the user's preferences over values for the attribute given every combination of parent values. The theory works on the concept of *ceteris paribus* (all else being equal), where preferences are defined and accepted as being true, given the conditions, all else equal. Let  $X$ ,  $Y$  and  $Z$  (non-empty) partition the set of attributes.  $X$  is said to be *conditionally preferentially independent* of  $Y$  given  $Z$  if, for any assignments  $x_1, x_2, y_1, y_2$  and  $z$  to those sets of attributes:

$$x_1 y_1 z \preceq x_2 y_1 z \text{ iff } x_1 y_2 z \preceq x_2 y_2 z$$

Thus if  $Z$  is the set of parent attributes of  $X$  (i.e. the conditions imposed on the preferences in  $X$ ), then given the conditions, preferences in  $X$  are said to hold *ceteris paribus*, meaning that values for attribute values for  $Y$  are irrelevant. We employ the *ceteris paribus* assumption in our model as well.

For example, suppose that there are four attributes,  $A, B, C$ , and  $D$ , and that each attribute has binary values ( $a$  and  $\bar{a}$  are values for attribute  $A$ ,  $b$  and  $\bar{b}$  for  $B$ , etc.); and that attribute  $A$  has no parent,  $A$  is the parent of  $B$ , and  $B$  is the parent of  $C$  and  $D$ . Suppose that the conditional preferences are as follows:

$$\begin{aligned} A : & \bar{a} \succ a && (\bar{a} \text{ is preferred over } a) \\ B : & \bar{a} : \bar{b} \succ b && (\text{given } \bar{a}, \bar{b} \text{ is preferred over } b) \\ & a : b \succ \bar{b} && (\text{given } a, b \text{ is preferred over } \bar{b}) \\ C : & \bar{b} : \bar{c} \succ c && (\text{given } \bar{b}, \bar{c} \text{ is preferred over } c) \\ & b : c \succ \bar{c} && (\text{given } b, c \text{ is preferred over } \bar{c}) \\ D : & \bar{b} : d \succ \bar{d} && (\text{given } \bar{b}, d \text{ is preferred over } \bar{d}) \\ & b : \bar{d} \succ d && (\text{given } b, \bar{d} \text{ is preferred over } d) \end{aligned}$$

The CP-network is shown in Figure 1.

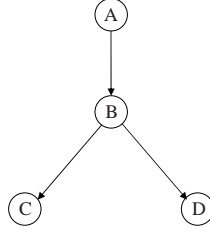


Fig. 1. An example CP-net

A dominance checking algorithm is then used to determine whether one outcome is preferred over another. The idea here is that preferences higher in the CP-net are more important than those that are lower. That is, if outcome  $o$  has a violation in the user's preference for attribute  $X$  (i.e. the less preferred value is present), and outcome  $o'$  has a violation in attribute  $X'$ , then if  $X$  is an ancestor of  $X'$  in the CP-net, then  $o$  is preferred over  $o'$ . In the example,  $\overline{a}\overline{b}\overline{c}\overline{d}$  has a violation in  $D$  and  $abcd$  has a violation in  $A$ . Since  $A$  is an ancestor of  $D$ , the violation in  $abcd$  is more damaging than the violation in  $\overline{a}\overline{b}\overline{c}\overline{d}$ , and thus  $\overline{a}\overline{b}\overline{c}\overline{d}$  is preferred over  $abcd$ . This can be shown by a sequence of “flips”:

$$\begin{aligned}
 abcd &\prec \overline{a}bcd \text{ (since } \overline{a} \succ a, a \text{ is flipped to } \overline{a}) \\
 \overline{a}bcd &\prec \overline{a}\overline{b}cd \text{ (since } \overline{a} : \overline{b} \succ b, b \text{ is flipped to } \overline{b}) \\
 \overline{a}\overline{b}cd &\prec \overline{a}\overline{b}\overline{c}d \text{ (since } \overline{b} : \overline{c} \succ c, c \text{ is flipped to } \overline{c})
 \end{aligned}$$

### 3 Conditional Outcome Preference Networks

In this section we define a structure for representing the specified preferences in such a way that new preferences that can be directly inferred will be immediately evident. The structure is a directed graph that represents preferences over the set of outcomes, and is referred to as a *Conditional Outcome Preference Network (COP-network)*. The main aim in using the network is to (1) determine whether one outcome is preferred over another, and (2) estimate utilities for the entire set of outcomes.

#### 3.1 Creating an Initial COP-Network

Users can specify preferences in many formats. For example, a preference could be specified as a comparison of values from the same attribute or across different attributes, with or without condition, and over two or more than two values. While the CP-networks described in Section 2 are restricted to representing preferences over values within a single attribute, COP-networks can also handle preferences across different attributes.

To create an initial COP-network, each given preference is transformed to a standard format referred to as a *preference rule*. A preference rule  $\overline{a_1} \succ \overline{a_2}$

for a set  $A$  of attributes is defined as a specification that represents that one assignment  $\bar{a}_1$  to the attributes in  $A$  is preferred over another assignment  $\bar{a}_2$ . Before the COP-network is constructed, all preferences specified by the user are transformed to preference rules. For example, the two conditional preferences  $\bar{a} : b\bar{c} \succ \bar{b}c$  and  $a : \bar{b}c \succ b\bar{c}$  would be transformed into the two preference rules  $\bar{a}b\bar{c} \succ \bar{a}\bar{b}c$  and  $a\bar{b}c \succ ab\bar{c}$ .

A COP-network is represented by a directed graph, where every outcome is represented by a node, and for nodes  $n$  and  $n'$  representing outcomes  $o$  and  $o'$ , respectively, if  $n$  is a proper ancestor of  $n'$  then  $o$  is necessarily preferred over  $o'$ , given the specified preferences and the *ceteris paribus* assumption. Initially, for every specified preference  $o \succ o'$ , an arc is inserted from the node representing  $o$  to the node representing  $o'$ . In subsequent sections, we discuss consistency checking and removal of redundant edges. Such a graph, without consistency checking and reduction, is referred to as an *initial* COP-network.

**Example 3.1.** Suppose that there is a set  $\{A, B, C\}$  of attributes, and that each attribute has binary values ( $a$  and  $\bar{a}$  are values for attribute  $A$ ,  $b$  and  $\bar{b}$  for  $B$ ,  $c$  and  $\bar{c}$  for  $C$ ), and that there are the following preferences:

$$\bar{a} \succ a, \quad \bar{b} \succ b, \quad \bar{c} \succ c, \quad a\bar{b} \succ \bar{a}b, \quad \bar{a} : b\bar{c} \succ \bar{b}c, \quad a : \bar{b}c \succ b\bar{c}$$

To structure a COP-network with the above preferences, all feasible outcomes are listed:  $\bar{a}\bar{b}\bar{c}$ ,  $\bar{a}\bar{b}c$ ,  $\bar{a}b\bar{c}$ ,  $\bar{a}bc$ ,  $a\bar{b}\bar{c}$ ,  $a\bar{b}c$ ,  $ab\bar{c}$ ,  $abc$ . Next, preference rules as dictated by the given preferences are applied to the outcomes, and a set of preferences over the outcomes is generated. For example, by applying the preference rule  $\bar{a} \succ a$ , we determine that outcome  $\bar{a}bc$  is preferred over outcome  $abc$ . The final step is to build a directed graph by creating a node for every outcome and adding a directed edge from node  $n_i$  to node  $n_j$  if the preference  $o_i \succ o_j$  holds for the corresponding outcomes. The resulting graph is shown in Figure 2, where Table 1 denotes which outcome is represented by each node.

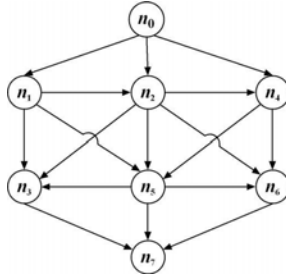
**Table 1.** Node representation for Figure 2. Bar values are removed (e.g.  $\bar{a}\bar{b}\bar{c} \Rightarrow a$ ).

Node:	$n_0$	$n_1$	$n_2$	$n_3$	$n_4$	$n_5$	$n_6$	$n_7$
Outcome:	$\phi$	$a$	$b$	$ab$	$c$	$ac$	$bc$	$abc$

### 3.2 Consistency

In decision making, an outcome cannot be preferred over itself. For any given set of preferences, it can be determined whether an outcome is preferred over itself by building an initial COP-network and checking whether there is a cycle in the network. An outcome is preferred over another outcome if there is a path from a node for the first outcome to another node for the second outcome. An initial COP-network is said to be consistent if and only if there is no outcome that is preferred over itself - *i.e.*, if and only if the network is acyclic. If a COP-network corresponding to a given set of preferences is found to have a cycle, then the user must be consulted in order to correct the inconsistency.



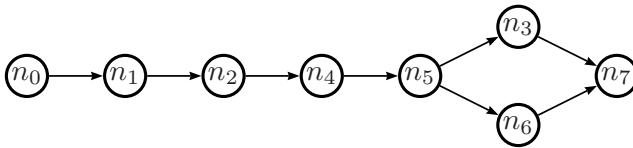


**Fig. 2.** The initial COPN

### 3.3 Reducing an Initial COP-Network

For nodes  $n_i$ ,  $n_j$  and  $n_k$  in an initial COP-network, if there are two paths  $n_i \rightarrow n_j \rightarrow \dots \rightarrow n_k$  and  $n_i \rightarrow n_k$ , the second path (i.e. the arc from  $n_i$  to  $n_k$ ) is not necessary since preferences that are reflected by the first path include the preference that the second path reflects. Thus, the arc  $(n_i, n_k)$  is said to be *redundant* and can be removed. This results in a transitive reduction of the initial COP-network. The aim of reducing the network is to make it easy to compute a utility function based on the network.

The graph from Figure 2 can be reduced to the graph shown in Figure 3.



**Fig. 3.** The reduced COP-network

### 3.4 Preference Checking

If each outcome is associated with an offer in a negotiation, the ability to show that one outcome is preferred over another should help the decision making in the negotiation. In a COP-network, the outcome corresponding to a node is preferred over the outcome associated with any proper descendant. For any pair of outcomes, preference checking is quite easy and efficient. For example, for any pair of outcomes  $o_i$  and  $o_j$  with corresponding nodes  $n_i$  and  $n_j$ :

- If  $n_i$  is a proper ancestor of  $n_j$ ,  $o_i$  is preferred over  $o_j$  ( $o_i \succ o_j$ );
- If  $n_i$  is a proper descendant of  $n_j$ ,  $o_j$  is preferred over  $o_i$  ( $o_j \succ o_i$ );
- If  $n_i$  is neither an ancestor nor a descendant of  $n_j$ , neither of  $o_i$  and  $o_j$  is known to be preferred over the other

Consider Example 3.1. Outcome  $o_0$  is preferred over outcome  $o_1$  since  $n_0$  is the parent of  $n_1$ . Outcome  $o_1$  is preferred over outcome  $o_7$  since  $n_1$  is an ancestor of  $n_7$ . Neither of outcomes  $o_3$  and  $o_6$  is known to be preferred over the other since  $n_3$  is neither an ancestor nor a descendant of  $n_6$ .

## 4 COPN Utility Functions

In addition to obtaining a set of preferences from the user during the elicitation stage, our model also allows for querying about specific utilities for outcomes. This can be done by asking standard gamble questions (see Keeney and Raiffa [8]), or by initially treating utility for an outcome as a random variable from a known distribution, and querying the user to sufficiently reduce the uncertainty in the utility estimate (see Chajewska et al. [5]), to cite some examples. Note that there will always be at least two outcomes for which utility is known, since we employ the convention of assigning a utility of 1 to the most preferred outcome (the topmost node in the network), and a utility of 0 to the least preferred outcome (the bottommost node in the network). Based on the COP-net derived from the specified preferences, and the partial utility function  $u : O' \rightarrow \mathbb{R}$  specifying utilities for a subset  $O'$  of outcomes, a utility function  $\hat{u}$  over the entire set  $O$  is produced. This is done in such a way as to preserve the preference ordering specified by the COP-net. Specifically, let  $n$  and  $n'$  represent outcomes  $o$  and  $o'$ . If  $n$  is a proper ancestor of  $n'$ , then  $\hat{u}(o) > \hat{u}(o')$ .

This section demonstrates three techniques for computing  $\hat{u}$ : The Bounded method, the Random-Path method and the Longest-Path method. Each method is then tested for accuracy against an existing method.

### 4.1 The Bounded Method

The Bounded method computes the utility by setting upper and lower bounds for each outcome  $o$  for which utility is unknown, and assigns the average of these bounds as the utility value. Let  $n$  represent  $o$  in the tree, let  $O_k$  be the set of outcomes for which the utility is known, and let  $O_a, O_d \subseteq O_k$  be the set of outcomes represented by ancestors and descendants of  $n$ , respectively. Then the Bounded method computes the utility estimate  $\hat{u}_B$  as

$$\hat{u} = \frac{\min\{u(o') \mid o' \in O_a\} + \max\{u(o') \mid o' \in O_d\}}{2} \quad (1)$$

By selecting the value that lies in the middle of the possible range, the bounded method produces utilities that are, in most cases, not too far from the true utilities. However, when there are paths of outcomes in the COP-net for which the preference ordering is known, if the utilities for the outcomes have the same upper and lower bounds the Bounded method will assign the same utility to each outcome. The next two methods overcome this drawback by assigning utilities that preserve preference orderings.

## 4.2 The Random-Path Method

Given a COP-network and a set of known utilities, the Random-path technique randomly selects a path of outcomes in the network for which utilities are unknown, and assigns utilities to those outcomes in such a way that preserves this preference ordering. Formally, let  $p = (o_1, o_2, \dots, o_n)$  be a path in the network. This path is a candidate for selection if (1)  $\hat{u}$  is known for  $o_1$  and  $o_n$ , and  $\hat{u}$  is unknown for all other outcome nodes on  $p$ , (2) for all paths satisfying (1),  $\hat{u}(o_1)$  is minimal, and (3) for all paths satisfying (1) and (2),  $\hat{u}(o_n)$  is maximal. Requirements (2) and (3) ensure consistency in the utilities in the graph<sup>1</sup>. Once a suitable path  $p$  has been selected, the utility  $\hat{u}$  is assigned for each outcome on  $p$ , decreasing from  $o_1$  to  $o_n$ , by

$$\hat{u}(o_i) = \hat{u}(o_n) + \frac{(n-i)(\hat{u}(o_1) - \hat{u}(o_n))}{n-1} \quad (2)$$

For example if  $p$  consisted of four outcomes with  $\hat{u}(o_1) = 0.8$  and  $\hat{u}(o_4) = 0.2$ , then  $\hat{u}(o_2)$  and  $\hat{u}(o_3)$  would be assigned utilities of 0.6 and 0.4, respectively.

The process of selecting paths at random and assigning utilities in this way continues until all outcomes are considered.

## 4.3 The Longest-Path Method

The Longest-Path method works in much the same manner as the Random-Path method, except that the longest acceptable candidate path is always selected. Utilities for outcomes are assigned according to Equation 2. Path selection continues until all outcomes have been considered.

## 4.4 A Simple Example

Consider the COP-network in Figure 4 containing 6 nodes, where each node  $n_i$  represents outcome  $o_i$ . Initially let  $u(o_1) = 0.82$  and  $u(o_6) = 0.1$ . Each of the three techniques described above computes  $\hat{u}$  for outcomes  $o_2$  to  $o_5$  as follows:

**Bounded:** Since the upper bound for all outcomes  $o_2$  to  $o_5$  is 0.82 and the lower bound is 0.1,  $\hat{u}(o_i) = \frac{0.82+0.1}{2} = 0.46$  for all  $i = 2$  to 5.

**Random-Path:** Random-Path will randomly begin with one of two paths:  $p_1 = (o_1, o_2, o_3, o_5, o_6)$  or  $p_2 = (o_1, o_4, o_5, o_6)$ .

1. If  $p_1$  is chosen first, then  $\hat{u}(o_2) = 0.64$ ,  $\hat{u}(o_3) = 0.46$  and  $\hat{u}(o_5) = 0.28$ . Next, path  $(o_1, o_4, o_5)$  is chosen and  $\hat{u}(o_4) = 0.55$ .
2. If  $p_2$  is chosen first, then  $\hat{u}(o_4) = 0.58$  and  $\hat{u}(o_5) = 0.34$ . Next, path  $(o_1, o_2, o_3, o_5)$  is chosen and  $\hat{u}(o_2) = 0.66$  and  $\hat{u}(o_3) = 0.5$ .

**Longest-Path:** Longest-Path will choose  $p_1 = (o_1, o_2, o_3, o_5, o_6)$  first (since it is the longer of  $p_1$  and  $p_2$ ), and compute utilities as in point 1 in Random-Path above.

---

<sup>1</sup> Refer to Chen [6] for more on ensuring consistency in path selection.

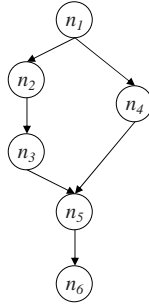


Fig. 4. An example COP-net for computing utilities

## 5 Analysis

### 5.1 Accuracy Testing

There are three techniques developed in this paper and discussed in previous sections. Given a set of preferences and given utilities for some of the possible outcomes, each technique constructs a COP-network and develops a utility function to predict all unknown utilities. Experiments were run to compare the accuracy of these three techniques as well as a previously developed technique for determining utilities, which we refer to as the *additive utility*. This technique, which is used by the “MONOLOGUE” automated negotiation system [4], handles interdependencies among attribute values that result from the specified conditional preferences by modifying the amount of utility that each attribute value contributes in a given outcome. For example, if an attribute value  $a$  is considered less desirable when attribute value  $b$  is present, then  $a$  contributes less utility to an outcome including  $b$  than it would to an outcome not including  $b$ . The overall utility for an outcome is then the sum of these modified utilities.

To test the accuracy of the algorithms, test cases were generated for different numbers of attributes and different numbers of conditional preferences. Tests were then run on these cases to determine how accurately the techniques could estimate a simulated user’s true utilities for all outcomes, given a small number of preferences and known utilities. In this experimentation, there were 33 test cases. The numbers of attributes ranged from 4 to 9. The numbers of conditional preferences ranged from 0 to 7.

For each test case, several sets of user preferences were generated, giving a large number of different test COP-networks. For each of these COP-networks, 10,000 trials were run. In each trial, a set of true utilities was generated to be consistent with the given preferences. Each of the four techniques (Bounded, Random-Path, Longest-Path and Additive) was then tested, to determine how accurately it could estimate the true utilities.

In order to determine the accuracy of the four techniques, two measures were considered: the differences between computed utilities and true utilities and the standard errors of those differences. For each test case, each technique computed

the utility for each outcome, and the difference between the computed utility and the true utility was noted. The winning technique was determined for each of the following criteria:

1. **Total difference winner:** The technique with the lowest difference for an outcome is viewed as having the best ability to predict utility for that outcome in the particular test case. This technique is deemed the *difference winner* for the outcome. For all test cases, the technique deemed the difference winner the most often is viewed as having the best ability to predict utility. This technique is deemed the *total difference winner*.
2. **Difference mean winner:** For all test cases, the technique with the lowest *mean* of differences over all outcomes is deemed the *difference mean winner*.
3. **Total standard error winner:** For each trial, the standard error over the set of estimated utilities is measured. The technique with the lowest standard error is deemed the *standard error winner* for the trial. For all test cases, the technique deemed the standard error winner the most often is viewed as having the best ability to predict utility. This technique is deemed the *total standard error winner*.
4. **Standard error mean winner:** For all test cases, the technique with the minimal *mean* of standard errors over all outcomes is deemed to be the *standard error mean winner*.

The accuracy of predicting utility of the four techniques is evaluated by considering the *total difference winner*, the *difference mean winner*, the *total standard error winner*, and the *standard error mean winner*. Table 2 shows the number of times each technique was the winner for each of these four criteria. Clearly, the Longest-Path technique is shown to most accurately predict utility regardless of the numbers of attributes and conditional preferences.

**Table 2.** Experimental results

Technique	Total difference winner	Difference mean winner	Standard error winner	Standard error mean winner
Bounded	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Random-Path	1 (3%)	6.5 (20%)	1 (3%)	5 (15%)
Longest-Path	23 (70%)	25.5 (77%)	32 (97%)	28 (85%)
Additive	9 (27%)	1 (3%)	0 (0%)	0 (0%)

## 5.2 Discussion on Running Time

Since a COP-net contains a node for every possible outcome, run-time for building and traversing the tree is very expensive in the worst case. Let  $n$  denote the number of attributes. If attributes are binary-valued, there are  $2^n$  outcomes. Testing showed that algorithms for computing utilities began to slow significantly at  $n = 15$ . We envision that, in most practical applications of the technology, 15 attributes is more than sufficient. For example, when negotiating which items

will be exchanged in a privacy scenario, or which options will be included in a car, it is difficult to imagine scenarios where both parties have enough concern over so many variables that more than 15 would need to be negotiated. However, in cases where significantly more items are involved, the COP-net can be divided into two or more sub-networks. This is done by partitioning the set of attributes such that dependent attributes are grouped together, and independent attributes are separated. A COP-network is then built for each group. Each such group is unlikely to consist of more than 15 attributes. Utilities can then be computed for outcomes in these COP-nets independently, and a multi-attribute utility function can be used to determine utilities for complete outcomes.

## 6 Conclusions and Related Work

In this paper, a graphical model referred to as a Conditional Outcome Preference Network (COP-network) is described. Using this model, techniques are developed to infer user preferences and utilities over all possible outcomes, given a small set of known preferences and utilities. Previous preference networks (such as CP-networks [1]) have handled only preferences specified over values for a particular attribute. In this paper, techniques are developed that can infer preferences over outcomes when user preferences are specified for values across attributes as well. As in previous techniques, conditional preferences are also handled. Efficient algorithms have been presented for checking the consistency of a COP-network and for using a given network to determine the user's preferences between any two possible outcomes. Three techniques are presented for estimating utilities for outcomes in the COP-net: Bounded, Random-Path and Longest-Path. Experiments show that the Longest-Path technique achieves the best results of the three, and also outperforms an existing technique.

Preference elicitation is becoming an increasingly popular topic for researchers working in the areas of agents and electronic commerce. Boutilier et al. [3] propose a minimax regret-based approach to preference elicitation. Given a decision problem, choices are made that the user would regret the least should an adversary choose the utility function consistent with the elicited preferences. If regret is higher than some threshold, then more querying is necessary. Determining such a consistent utility function is difficult, especially when conditional preferences exist, so perhaps our work can complement this. Other works on utility elicitation, such as those by Chajewska et al. [5] and Haddawy et al. [7], demonstrate effective ways to estimate utilities based on data obtained on other individuals' preferences or utilities over outcomes. Our work differs from these as we assume that no such data exists.

## 7 Future Work

For future work, a COP-network capturing a user's known preferences and a set of estimated utilities could be used to make decisions about which preference elicitation questions to ask next. If two nodes in the graph have the property

that neither is an ancestor of the other, then it would be reasonable to ask the user a preference elicitation question with the goal of determining which outcome is preferable. However, learning the user's preference over some pairs of outcomes might be more informative than other pairs. For example, due to the structure of the graph, learning that the user prefers outcome  $o_1$  over outcome  $o_2$  ( $o_1 \succ o_2$ ) might also tell the agent that  $o_3 \succ o_2$  and that  $o_1 \succ o_4$ . This then has the potential to be a more useful question than one that provides no such additional preference information.

The approach to this problem will include experimenting with graph-theoretic methods to find a set of candidate edges corresponding to potential preference elicitation questions, and then evaluating the anticipated effect of learning the answer to each of these questions on the expected utility of the agent's strategy (perhaps a negotiation strategy to be used in negotiations with an opposing agent). The question perceived to yield the highest increase in expected utility would be the next question chosen.

Another possible direction for future work is to investigate the feasibility of reducing the space consumption of the current COP-network model by attempting to modify the network so that it contains a node for each attribute rather than each outcome.

## References

1. C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research (JAIR)*, 21:135–191, 2004.
2. C. Boutilier, R. I. Brafman, C. Geib, and D. Poole. A constraint-based approach to preference elicitation and decision making. *AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, pages 19–28, 1997.
3. C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Regret-based utility elicitation in constraint-based decision problems. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 929–934, Edinburgh, Scotland, 2005.
4. S. Buffett, L. Comeau, M. W. Fleming, and B. Spencer. Monologue: A tool for negotiating exchanges of private information in e-commerce. In *Third Annual Conference on Privacy, Security and Trust (PST05)*, pages 79–88, 2005.
5. U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *AAAI-00*, pages 363–369, Austin, Texas, USA, 2000.
6. S. Chen. Reasoning with conditional preferences across attributes. Master's thesis, University of New Brunswick, 2006.
7. P. Haddawy, V. Ha, A. Restificar, B. Geisler, and J. Miyamoto. Preference elicitation via theory refinement. *Journal of Machine Learning Research*, 4:313–337, 2003.
8. R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley and Sons, Inc., 1976.

# Path Propagation for Inference in Bayesian Networks

Dan Wu and Liu He

School of Computer Science  
University of Windsor  
Windsor Ontario  
Canada N9B 3P4

**Abstract.** Although (probabilistic) inference in Bayesian networks has been well studied, the recent trend on extending Bayesian networks to model large and complex domains imposes new challenges on inference. In this paper, we suggest a method called path propagation that addresses these new challenges. The experimental results indicate that the proposed method achieves better performance than conventional method, especially for large Bayesian networks.

## 1 Introduction

The Bayesian network (BN) model is a probabilistic graph model that has been successfully developed and applied in various domains for uncertainty management [6]. It abstracts a problem domain using a set  $U$  of random variables and a *directed acyclic graph* (DAG) to encode the conditional independency information among variables in  $U$ . One of the most important services that a BN provides is inference (or probabilistic inference), which simply means calculating posterior probability  $p(x|\mathbf{E} = e)$  for a variable  $x \in U$  given that variables in  $\mathbf{E}$  are taking a specific value  $e$ . Computing  $p(x|\mathbf{E} = e)$  is also called a *query* in this paper. Various research has been carried out on designing and implementing algorithms for performing inference [2], for instance, the renowned *global propagation* (GP) method originally developed in [4, 3].

Although quite a successful model for uncertainty management, researchers have noticed that the BN model has inherent deficiencies in its modeling capacity for large and complex domains. Very recently, research has been done to extend the BN model to handle such domains, for instance, the *multiply sectioned Bayesian network* (MSBN) model and the *object-oriented Bayesian network* (OOBN) model. Both models focus more on providing methodologies for modeling large and complex domains than developing completely new methods for inference. In fact, it was suggested that one can transform a MSBN or OOBN to a single BN on which the GP method can be applied [1].

The GP method performs quite well on many small or medium BNs in practice. However, the idea of applying the GP method to a large BN (for instance, a BN converted from a MSBN or OOBN) for inference presents new challenges.



Adverse scenarios could arise in the process of inference in large BNs. For example, the response time of inference could be slow, or inference using the GP method is not possible at all because of the size of the BN. One of our recent experiments shows that Hugin fails to operate on a network with about 1300 nodes, reporting an out of memory error message on a P4 machine with 512 MB memory and normal load.

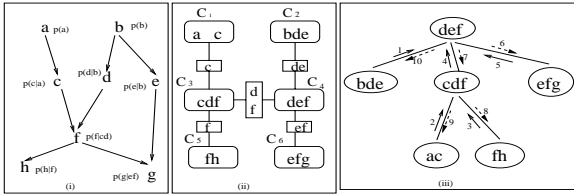
In this paper, we suggest an on-demand thrifty method designed for inference in large BNs. This approach is based on the GP method but with novel features tailored for large BNs. By identifying common query patterns and making full use of known properties of the GP method, in the proposed approach, the computation needed for answering a query  $p(x|\mathbf{E} = e)$  is *on-demand* and *thrifty* in the sense that only the necessary minimal computation is carried out to obtain  $p(x|\mathbf{E} = e)$ .

The paper is organized as follows. In Section 2, we review the GP method for inference. In Section 3, we discuss the new challenges of inference in large BNs. In Section 4, we present the proposed method for inference that addresses the challenges. We show the experimental results in Section 5 and conclude the paper in Section 6.

## 2 Global Propagation for Inference

A *Bayesian network* defined over a set  $U$  of variables, written as  $(\mathcal{D}, \mathcal{P})$ , is a probabilistic graphical model where  $\mathcal{D}$  denotes a DAG and  $\mathcal{P}$  denotes a set of *conditional probabilistic distributions* (CPDs), such that each node  $x$  in  $\mathcal{D}$  corresponds one-to-one to a variable in  $U$  and each node is also associated one-to-one with a CPD  $p(x|\pi_x)$  in  $\mathcal{P}$ , where  $\pi_x$  denotes the parents of  $x$  in  $\mathcal{D}$ . The product of the CPDs in  $\mathcal{P}$  defines a *joint probability distribution* (JPD) over  $U$  as  $p(U) = \prod_{x \in \mathcal{D}} p(x|\pi_x)$  [6].

One of the standard methods for performing inference is the so-called *global propagation* (GP) method originally developed in [4]. Since GP is well understood, we refer readers to [2] for a complete exposition. We will only describe relevant background of the GP method that will be used in the paper using an example.



**Fig. 1.** (i) The Asia BN  $\mathcal{D}$ . (ii) The junction tree transformed from the Asia BN in (i). (iii) GP is performed on the junction tree with  $C_4$  as root; separators are omitted for simplicity and clarity.

Consider the Asia BN [4] ( $\mathcal{D}$ ,  $\mathcal{P}$ ) defined over  $U = \{a, b, c, d, e, f, g\}$ . Its DAG  $\mathcal{D}$  is shown in Figure 1 (i). For simplicity, we assume all variables in  $U$  are binary. In order to perform GP, the DAG has to be first transformed into a junction tree, shown in Figure 1 (ii), through the *moralization* and *triangulation* processes. The round rectangles in the junction tree represent cliques and they are  $C_1 = ac$ <sup>1</sup>,  $C_2 = bde$ ,  $C_3 = cdf$ ,  $C_4 = def$ ,  $C_5 = fh$ , and  $C_6 = efg$ . The (smaller) rectangles represent separators, i.e., the intersections of two adjacent cliques and they are  $S_1 = c$ ,  $S_2 = de$ ,  $S_3 = df$ ,  $S_4 = f$ , and  $S_5 = ef$ . Along with the graphical transformation from the DAG to the junction tree, each CPD  $p(x|\pi_x)$  in  $\mathcal{P}$  is also assigned to a unique clique  $C_i$  if  $\{x\} \cup \pi_x \subseteq C_i$  to form an associated clique potential (function) denoted  $\psi_i(C_i)$ . The potential  $\psi_i(C_i)$  is defined as the product of all CPDs assigned to clique  $C_i$  or 1 if no CPD is assigned to  $C_i$  at all. Each separator  $S_{ij}$  connecting two cliques  $C_i$  and  $C_j$  in the junction tree is also associated with a separator potential  $\psi_{ij}(S_{ij})$  with initial value 1. A potential can be represented as a table [7]. The size of a clique (potential) usually refers to the number of entries (rows) in the associated potential table.

The GP method is performed on the junction tree by picking an arbitrary clique as root and then passing messages inward and outward with respect to the chosen root. Passing a message between two cliques is the core operation of the GP method. Consider two adjacent cliques  $C_i$  and  $C_j$  with separator  $S_{ij} = C_i \cap C_j$ . Clique  $C_i$  passing a message to clique  $C_j$  means the following calculation:

$$\psi_j(C_j) = \psi_j(C_j) \cdot \frac{\sum_{C_i - S_{ij}} \psi_i(C_i)}{\psi_{ij}(S_{ij})}. \quad (1)$$

In Figure 1 (iii), clique  $C_4$  was picked as root. The solid arrows indicate inward pass of message passing, each clique potential passing a message to its neighbor towards the root, starting from the leaf cliques. The dashed arrows indicate outward pass of message passing, each clique potential passing a message to all its neighbors away from the root, starting from the root clique. The objective of GP is to compute the marginal distribution for each clique and separator in the junction tree. After the GP finishes, the clique potentials  $\psi(C_i)$  have now been turned into marginals  $p(C_i)$ . At this stage, the probability  $p(x)$  for any  $x \in U$  can be easily computed.

The notion of *evidence* means that some variables in  $U$  are taking specific values from their respective domains. For instance, if variable  $a$  takes the value 0 and variable  $c$  takes the value 1, then we use  $\mathbf{E}$  to denote this evidence and use  $v(\mathbf{E})$  to denote variables occurring in  $\mathbf{E}$ , i.e.,  $\{a, c\}$ . Two pieces of evidence  $\mathbf{E}$  and  $\mathbf{E}'$  are *contradicting* if there exists a variable  $x$  such that  $x \in v(\mathbf{E})$  and  $x \in v(\mathbf{E}')$  but  $x$  are taking different values in  $\mathbf{E}$  and  $\mathbf{E}'$ . Otherwise,  $\mathbf{E}$  and  $\mathbf{E}'$  are *compatible*.

With an evidence  $\mathbf{E}$  observed, we are interested in computing the posterior probability  $p(x|\mathbf{E})$  for variable  $x \in U$  of interest, to be compared with its prior probability  $p(x)$ . In order to compute  $p(x|\mathbf{E})$ , the evidence  $\mathbf{E}$  needs to be first

<sup>1</sup> By  $ac$ , we mean  $\{a, c\}$ , etc.

incorporated in some clique potential  $\psi_i(C_i)$  if  $v(\mathbf{E}) \subseteq C_i$  and the GP method has to be applied again on the junction tree. The objective of reapplying GP on a junction tree with evidence observed (and incorporated) is to compute the *updated* marginals for each clique (and separator). Note the updated marginal is in fact the probability distribution for the variables in the clique *conjoint* with the evidence  $\mathbf{E}$ . For instance, if evidence  $\mathbf{E}$  is  $a = 0$ , then after reapplying the GP method, we obtain  $p(C_i, \mathbf{E})$  for each clique. At this stage, the posterior probability  $p(x|\mathbf{E})$  for any  $x \in U$  is also readily available, and can be compared with  $p(x)$  to see how the evidence changes one's belief on  $x$ .

To summarize, applying the GP method to a junction tree *with no evidence observed* results in marginal distributions for cliques (and separator) computed; applying the GP method to a junction tree *with evidence observed and incorporated* results in updated marginal distributions computed for cliques (and separator). The application of the GP method is in *full* scale in the sense that all the cliques in the junction tree are involved in the propagation process.

### 3 Inference in Large BNs – New Challenges

Although the BN model and the GP method for inference are quite successful in many real applications with BNs of small or medium size, researchers are now aiming at modelling large and complex domains. For instance, the OOBN and MSBN model. It was suggested to perform inference by converting a OOBN or MSBN into a single BN on which the GP method can be applied. However, it is important to note that since both MSBN and OOBN models are originally developed to model large and complex domains, the BN converted could be much larger than any existing BNs.

There are two concerns regarding inference using the GP method for large BNs in general. First, the effectiveness and efficiency of the GP method in practice are strongly related to the size of all the cliques in a junction tree. As the sizes of BNs increase, it is expected that the size of all the cliques in a junction tree will be much larger, and this concerns the performance of inference on large BNs. It is foreseeable that larger BNs will take much longer time to perform GP and answer queries. Secondly, the GP method involves inward and outward message passing performed on the *whole* junction tree, the upside of such a propagation scheme is that probability (or posterior probability when evidence is observed) is readily available for every variable once the GP method finishes its execution. However, in many applications, it is unreasonable to assume that users are interested in the probability (or posterior probability when evidence is observed) for each and every variable in the network, especially in large BNs. For instance, in the “Munin3” BN which has 1045 variables, if evidence observed, performing GP on this network could produce posterior probability for each of the 1045 variables, it is unlikely that the posterior probability for each variable in the network will be of interest to the user whenever new evidence is observed. In most cases, only the posterior probabilities of a few variables are of interest [7], that means much of the computation occurred when applying GP is totally wasted. In other

words, performing a full scale GP only for a few variables of interest is not an economic way to utilize computing resources.

## 4 Path Propagation in Large BNs

In response to the concerns just expressed, in this section, we present an on-demand thrifty propagation method called *path propagation* (PP). The path propagation method is based on the assumption that the GP method is applied in full scale *only once* on a junction tree with no evidence observed, and the marginals for cliques and separators in the junction trees are known thereafter. This assumption is justified by how a BN is used in practice. In real applications, one always needs to know the prior probability  $p(x)$  for a variable  $x$  of interest and compare it with posterior probability  $p(x|\mathbf{E})$  to see how the evidence  $\mathbf{E}$  affects one's belief on  $x$ .

We first introduce two theorems that lay the foundation for path propagation. Consider two adjacent cliques  $C_i$  and  $C_j$  in a junction tree with separator  $S_{ij} = C_i \cap C_j$ . Let  $p(C_i)$ ,  $p(C_j)$ , and  $p(S_{ij})$  be the marginals corresponding to  $C_i$ ,  $C_j$ , and  $S_{ij}$ , respectively. Suppose an evidence  $\mathbf{E}$  is observed such that  $v(\mathbf{E}) \subseteq C_i$ . We then have the following theorem.

### Theorem 1

$$p(C_j, \mathbf{E}) = p(C_j) \cdot p(\mathbf{E}|S_{ij}) = p(C_j) \cdot \frac{\sum_{C_i - v(\mathbf{E}) - S_{ij}} p(C_i, \mathbf{E})}{p(S_{ij})}. \quad (2)$$

**Proof:** By product rule,  $p(C_j, \mathbf{E}) = p(C_j) \cdot p(\mathbf{E}|C_j)$ . Since  $C_j = S_{ij} \cup (C_j - S_{ij})$ , it then follows  $p(C_j, \mathbf{E}) = p(C_j) \cdot p(\mathbf{E}|S_{ij}, C_j - S_{ij})$ . The structure of the junction tree dictates that  $\mathbf{E}$  is conditionally independent of  $C_j - S_{ij}$  given  $S_{ij}$  [7], namely,  $p(\mathbf{E}|S_{ij}, C_j - S_{ij}) = p(\mathbf{E}|S_{ij})$ . Eq. 2 then follows naturally.  $\square$

Theorem 1 indicates that in order to obtain the updated marginal  $p(C_j, \mathbf{E})$ , we need to compute  $p(\mathbf{E}|S_{ij})$ . However,  $p(\mathbf{E}|S_{ij}) = \frac{p(\mathbf{E}, S_{ij})}{p(S_{ij})}$ , in which the denominator  $p(S_{ij})$  is the marginal on  $S_{ij}$  and the numerator  $p(\mathbf{E}, S_{ij})$  is readily available from  $p(C_i)$  (note that  $v(\mathbf{E}) \subseteq C_i$ ) because of our assumption. Moreover, the calculation in Eq. (2) is in exactly the same form as  $C_i$  passing a message to  $C_j$  in Eq.(1), and we call the computation in Eq. (2) as that the clique  $C_i$  (with the updated marginal  $p(C_i, \mathbf{E})$ ) passes the evidence  $\mathbf{E}$  to clique  $C_j$  (to obtain the updated marginal  $p(C_j, \mathbf{E})$ ).

*Example 1.* Consider again the junction tree shown in Figure 1 (iii). Suppose we now observe an evidence  $\mathbf{E}$  that  $e = 0$  and we want to compute  $p(cdf, \mathbf{E})$ . By Theorem 1,  $p(cdf, \mathbf{E}) = p(cdf) \cdot p(\mathbf{E}|cdf)$ , in which  $p(cdf)$  is known and  $p(\mathbf{E}|cdf)$  (i.e.,  $p(e = 0|cdf)$ ) can be computed from  $p(def, \mathbf{E})$  (i.e.,  $p(def, e = 0)$ ).

It is not hard to see that Theorem 1 implies that clique  $C_i$  with its updated probability  $p(C_i, \mathbf{E})$  can always pass the evidence  $\mathbf{E}$  to its adjacent clique  $C_j$  to obtain the updated probability  $p(C_j, \mathbf{E})$ .

Consider a more general case with two adjacent cliques  $C_i, C_j$ , and the separator  $S_{ij}$ . Let  $p(C_i, \mathbf{E}_i)$  and  $p(C_j, \mathbf{E}_j)$  be the marginals on  $C_i$  and  $C_j$  respectively where  $\mathbf{E}_i$  and  $\mathbf{E}_j$  are compatible evidences. Suppose another evidence  $\mathbf{E}'$  compatible with both  $\mathbf{E}_i$  and  $\mathbf{E}_j$  is observed and  $v(\mathbf{E}') \subseteq C_i$ . We then have the following theorem.

**Theorem 2.** The clique  $C_i$  with its updated marginal  $p(C_i, \mathbf{E}_i, \mathbf{E}')$  passing the evidence  $\mathbf{E}'$  to clique  $C_j$  results in the updated marginal  $p(C_j, \mathbf{E}_j, \mathbf{E}')$ .

Theorem 2 can be similarly proved as Theorem 1. Theorem 1 and Theorem 2 paves the way for introducing path propagation.

We begin our discussion of path propagation by first observing some noticeable patterns in queries in many real applications. Imagine the following scenario, a doctor, when diagnosing a patient with the help of an expert system built on the BN model, can ask the patient to take one or two lab tests because the doctor is suspicious of a few diseases, say diseases  $x$  and  $y$ , that the patient may suffer. The returned lab results are evidence  $\mathbf{E}$ , and the doctor is interested in the posterior probabilities of diseases  $x$  and  $y$  given the evidence  $\mathbf{E}$ , namely,  $p(x|\mathbf{E})$  and  $p(y|\mathbf{E})$ . It happens that  $p(x|\mathbf{E})$  and  $p(y|\mathbf{E})$  do not differ too much from their priors  $p(x)$  and  $p(y)$ . This does not give the doctor enough confidence to conclude the diagnosis. By referring to some medical diagnosis repository, the doctor realizes that the patient may also suffer a rare life threatening disease  $w$ , and the doctor wants to know  $p(w|\mathbf{E})$  to see if the patient possibly has disease  $w$ . We formalize the pattern exhibited in this scenario as the following inference task.

**Definition 1.** Query Pattern 1 (multiple variables): Given evidence  $\mathbf{E}$ , compute the posterior probability for  $x_1, x_2, \dots, x_n$  given  $\mathbf{E}$ , namely, compute  $p(x_1|\mathbf{E}), \dots, p(x_n|\mathbf{E})$ .

Imagine another scenario in the same medical diagnosis setting. The doctor suspects that the patient is suffering disease  $x$  and the doctor asks the patient to do a lab test to confirm the suspicion. However, the posterior probability  $p(x|\mathbf{E})$  does not warrant the doctor to make a conclusive diagnosis until another lab test can double confirm the suspicion. The patient has no choice but to do another lab test, i.e., providing another evidence  $\mathbf{E}'$ . This time, the posterior probability  $p(x|\mathbf{E}, \mathbf{E}')$ , does warrant the doctor to make a conclusive diagnosis. We formalize the pattern exhibited in this scenario as the following inference task.

**Definition 2.** Query Pattern 2 (multiple evidences): Compute the posterior probability for a variable  $x$  given incrementally compatible evidences  $\mathbf{E}_1, \mathbf{E}_2, \dots, \mathbf{E}_n$ , namely, compute  $p(x_1|\mathbf{E}_1), p(x_1|\mathbf{E}_1, \mathbf{E}_2), \dots, p(x_1|\mathbf{E}_1, \dots, \mathbf{E}_n)$ .

In the following, we introduce path propagation for computing posterior probability in the above two patterns.

We first introduce a procedure to compute  $p(x|\mathbf{E})$  where  $v(\mathbf{E})$  is contained by a clique  $C$ . Recall that we have assumed that GP had been applied once and marginals on each clique and separator in the junction tree are now known.

PROCEDURE *Compute*( $x, \mathbf{E}$ )

Input: variable  $x$  and evidence  $\mathbf{E}$  such that  $v(\mathbf{E})$  is contained by clique  $C$ .

Output:  $p(x|\mathbf{E})$ .

```
{
  1: Identify a clique  $C'$  such that  $x \in C'$ .
  2: Find out a path  $(C_0, C_1, \dots, C_m)$  in the junction tree such that
      $C_0 = C$ ,  $C_m = C'$ , and  $(C_k, C_{k+1})$  is an edge in the junction tree,
      $k = 0, \dots, m-1$ .
  3: For  $k = 0$  to  $m-1$ , clique  $C_k$  passes the evidence  $\mathbf{E}$  to  $C_{k+1}$ .
  4: Compute  $p(x|\mathbf{E})$  from  $p(C', \mathbf{E})$ .
  5: Mark each clique  $C_i$ ,  $i = 0, \dots, m$ , with evidence  $\mathbf{E}$ , denoted  $C_i^{\mathbf{E}}$ .
  6: Return  $p(x|\mathbf{E})$ .
}
```

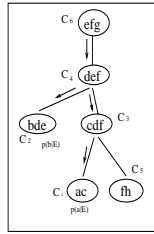
The above procedure for computing  $p(x|\mathbf{E})$  is based on Theorem 1. It first identifies a path in the junction tree connecting cliques  $C$  and  $C'$ ; it then begins to pass the evidence  $\mathbf{E}$  sequentially starting from clique  $C$ . Based on Theorem 1,  $p(C_i, \mathbf{E})$  for each clique  $C_i$  along the path is obtained. Note that the message passing occurring in the above procedure involves *only* the cliques along the path between  $C$  and  $C'$ . Only the cliques in the path are necessary for computing  $p(C', \mathbf{E})$ , from which  $p(x|\mathbf{E})$  can be obtained in Step 4. All the other cliques in the junction tree are irrelevant to the query  $p(x|\mathbf{E})$  and they are not involved in computing  $p(x|\mathbf{E})$ . As a side-effect, since  $p(C_i, \mathbf{E})$ ,  $i = 1, \dots, m-1$ , are all computed in Step 3, if  $y \in C_i$ , then  $p(y|\mathbf{E})$  is also readily available. In step 5, we mark each clique  $C_i$  as  $C_i^{\mathbf{E}}$  to indicate that the marginal on  $C_i$  is now conjoint with the evidence  $\mathbf{E}$ , i.e.,  $p(C_i, \mathbf{E})$ .

#### 4.1 Solving Pattern 1: Multiple Variables

A solution to the queries of pattern 1, namely, the multiple variables case, can be formulated based on the above procedure. Consider a fixed evidence  $\mathbf{E}$  and the task of computing  $p(x_1|\mathbf{E})$ ,  $\dots$ , and  $p(x_n|\mathbf{E})$  for a sequence of variables  $x_1, \dots, x_n$ . This can obviously be accomplished by calling the procedure *Compute*( $x, \mathbf{E}$ ) with fixed  $\mathbf{E}$  and different  $x_1, \dots, x_n$  variables in the sequence as the first argument. However, there is room for improving the efficiency as the following example shows.

*Example 2.* Consider again the Asia BN in Section 2, Figure 1. Suppose we want to know  $p(a|\mathbf{E})$  and  $p(b|\mathbf{E})$  given the fixed evidence  $\mathbf{E}$  denoting  $g = 1$ . The evidence is residing in clique  $C_6$ , and the variables  $a$  and  $b$  are residing in cliques  $C_1$  and  $C_2$ , respectively.

We invoke the procedure *Compute* first with the pair  $(a, \mathbf{E})$  as the arguments. According to the procedure, the path  $(C_6, C_4, C_3, C_1)$  in the junction tree is identified in step 2 shown in Figure 2. Note that every clique in the path will be marked to indicate that the marginal on that clique is conjoint with the evidence  $\mathbf{E}$  after the procedure finishes. In order to compute  $(b, \mathbf{E})$ , although we can find



**Fig. 2.** Compute  $p(a|\mathbf{E})$  and  $p(b|\mathbf{E})$  in pattern 1

a path  $(C_6, C_4, C_2)$ , however, it is important to note that marginals on  $C_6$  and  $C_4$  are already conjoint with  $\mathbf{E}$  (namely,  $p(C_6, \mathbf{E})$  and  $p(C_4, \mathbf{E})$ ) from previous application of the procedure on the pair  $(a, \mathbf{E})$  for computing  $p(a|\mathbf{E})$ . Passing the evidence  $\mathbf{E}$  from  $C_6$  to  $C_4$  has no effect. Therefore, passing evidence from  $C_6$  to  $C_4$  and then to  $C_2$  can be simplified as passing the evidence  $\mathbf{E}$  from  $C_4$  to  $C_2$ .

Example 2 indicates that in finding the path connecting the clique  $C'$  containing the variable whose posterior probability is of interest (say clique  $C_2$  in Example 2) and the clique  $C$  where the evidence is originally residing (say clique  $C_6$  in Example 2), we may possibly make use of the fact that the evidence may have been propagated to other cliques in the junction tree during previous computations such that the path found in step 2 between  $C$  and  $C'$  can be possibly shortened. In the path  $(C_0 = C, C_1, \dots, C_m = C')$  identified in step 2, we may examine each clique in the path in sequence and locate the *last* clique, say  $C_i$ ,  $0 \leq i \leq m$ , that has been marked in step 5 in the procedure *Compute* by the evidence  $\mathbf{E}$  before. In consequence, the original path  $(C_0 = C, C_1, \dots, C_m = C')$  can then be shortened to  $(C_i, \dots, C_m = C')$  which results in less computation for computing  $p(x|\mathbf{E})$  where  $x \in C'$ .

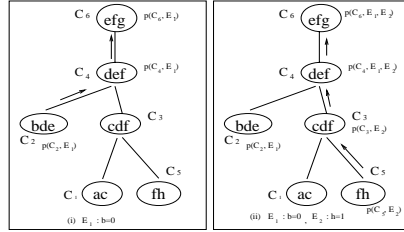
The final solution to query pattern 1, based on the above discussion, can be simply described as

- (1) identifying a path in the junction tree from a clique  $C$  whose marginal is conjoint with  $\mathbf{E}$  to the clique  $C'$  containing the variable of interest such that the length of such a path is the shortest, and
- (2) passing the fixed evidence  $\mathbf{E}$  along the path as in step 3 in the procedure *Compute*.

Imagine Pattern 1 in a large junction tree, answering the query  $p(x_i|\mathbf{E})$  using the above described method only involves a path in the junction tree and the evidence is passed along this path only. In other words, only a part of the original junction tree takes part in computing  $p(x_i|\mathbf{E})$  instead of a full scale GP. If the user is only interested in the posterior probabilities for a few variables given a fixed evidence, it is obvious that the proposed method saves a lot of computation compared with a full scale GP.

## 4.2 Solving Pattern 2: Multiple Evidences

The pattern of multiple evidence can be solved in a similar fashion. We explain the solution using an example first.



**Fig. 3.** Compute  $p(g|\mathbf{E}_1)$  and  $p(g|\mathbf{E}_1, \mathbf{E}_2)$  in pattern 2

*Example 3.* Consider the junction tree in Figure 2. Suppose the first evidence observed is  $\mathbf{E}_1$  which denotes  $b = 0$ , and we want to know  $p(g|\mathbf{E}_1)$ . The evidence is residing in clique  $C_2$ , and the variables  $g$  is residing in cliques  $C_6$ . The evidence  $\mathbf{E}_1$  can be passed as indicated by arrows in Figure 3 (i) from clique  $C_2$  to  $C_6$  via  $C_4$ . Note that the updated marginals on cliques  $C_4$  and  $C_6$  are  $p(C_4, \mathbf{E}_1)$  and  $p(C_6, \mathbf{E}_1)$  respectively once the procedure *Compute* finishes according to Theorem 2.

Suppose we further observed evidence  $\mathbf{E}_2$  which denotes  $h = 1$ , and we want to know  $p(g|\mathbf{E}_1, \mathbf{E}_2)$ . The evidence  $\mathbf{E}_2$  is residing in clique  $C_5$ . According to Theorem 2, we can pass the evidence  $\mathbf{E}_2$  towards the clique  $C_6$  which contains the variable  $g$  to obtain  $p(C_6, \mathbf{E}_1, \mathbf{E}_2)$ , from which  $p(g|\mathbf{E}_1, \mathbf{E}_2)$  can be finally computed. The evidence  $\mathbf{E}_2$  was passed as indicated by arrows in Figure 3 (ii) from clique  $C_5$  to  $C_6$  via  $C_3$  and  $C_4$ . Note that the updated marginals for each clique along the path connecting  $C_5$  and  $C_6$  are also indicated.

Example 3 gives rise to a solution to the multiple evidence scenario of inference as follow. We can

- (1) first pick the clique which contains the variable of interest as the root of the junction tree, and
- (2) whenever new (compatible) evidence  $\mathbf{E}_i$  is observed, we pass  $\mathbf{E}_i$  from the clique containing the evidence  $\mathbf{E}_i$  to the chosen root which contains the variable of interest.

The updated marginal on the root thus is always conjoint with all the evidence observed so far.

Imagine Pattern 2 in a large junction tree, in order to answer the queries  $p(x_i|\mathbf{E}_1)$ ,  $p(x_i|\mathbf{E}_1, \mathbf{E}_2)$ ,  $\dots$ ,  $p(x_i|\mathbf{E}_1, \dots, \mathbf{E}_m)$ , based on the above described method, one has to pass evidence  $\mathbf{E}_j$  from a clique contain  $v(\mathbf{E}_j)$  to the clique containing variable  $x_i$  for  $j = 1, \dots, m$  sequentially. This process involves only



$m$  paths in the whole junction tree and there is no need to perform a full scale GP which incurs much more computation than necessary for the query imposed.

To summarize, the salient feature of the proposed method is that (1) the computation occurred during the propagation is based on the query imposed by the users and it answers the query only, and (2) the computation for obtaining the posterior probability for the variables of interest is minimal in the sense that the propagation only involves those cliques that *have to* participate to produce the results. Such a thrifty method avoids a full scale GP, takes less time to answer queries, and waste less computational resources.

## 5 Experimental Results

In this section, we present experimental results that demonstrate the effectiveness and efficiency of path propagation. Both GP and path propagation are implemented in C language under Unix for comparison. The two implementations are identical to each other except the propagation part. As in [5], we use the number of binary arithmetic operations (additions, multiplications, and divisions) in GP and path propagation as the measurement of efficiency of the implementations.

The experiments are conducted on 4 Bayesian networks with increasing sizes. They are *car-ts.net* with 12 variables, *4sp.net* with 58 variables, *pigs.net* with 441 variables, and *munin2.net* with 1003 variables. Query patterns 1 and 2 are tested on each of the networks.

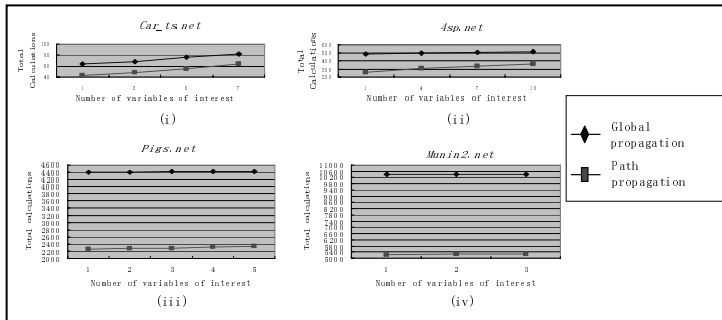
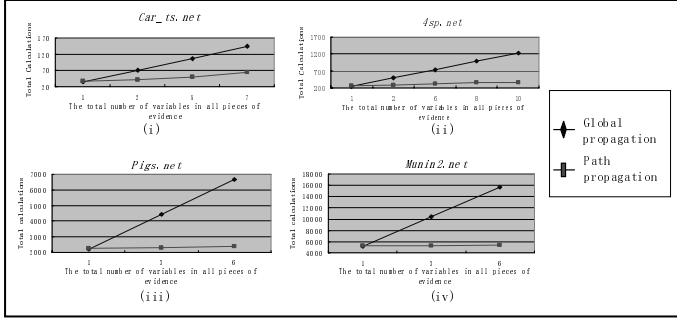


Fig. 4. Experimental Results for query pattern 1

Figure 4 shows the results of query pattern 1. Taking *4sp.net* as an example, it contains four different query sessions<sup>2</sup>, and the evidence used in each session is different. For each query session, the number of variables of interest is gradually increasing. For example, the first query session is for one variable; the second

<sup>2</sup> A query session of pattern 1(or 2) is a sequence of queries that satisfies the definition of query pattern 1(or 2).

is for 3 variables; the third is for 5 variables; the fourth is for 7 variables. All variables of interest in each query session are generated randomly. The X-axis represents the number of variables of interest in each query session, and the numbers in Y-axis represent the total number of calculations needed for both propagations.



**Fig. 5.** Experimental Results for query pattern 2

Figure 5 shows the results of query pattern 2. Taking *car\_ts.net* as an example, it contains one query session with increasing number of evidences (4 pieces of evidence in total). For example, at the beginning of the query session, there is only one piece of evidence; then the second piece of evidence comes, then the third and fourth pieces of evidence come. The evidences in the session are generated randomly. The X-axis represents the number of variables in all pieces of evidence in the query session, and the numbers in Y-axis represent the total number of calculations needed for both propagation.

The results in both figures obviously indicate that path propagation uses much less arithmetic operations than GP for inference. In Figure 5, the slope of the curve for path propagation is much smaller than that of GP which implies that the more the number of variables of interest, the greater the savings of needed arithmetic calculation. In Figure 4, the slope of the curve for path propagation is only marginally greater than that of GP. That means the two curves will intersect at some point. However, the x-axis value at the intersect point, which represents the number of variables of interest, will be a very large number. Since users of BNs are only interested in the probability of a few variables in the network, the fact that the arithmetic calculations for both path propagation and GP will be equal at some point does not really concern us. All these advantages of path propagation is simply because that the GP method has to be operated on the whole junction tree, while path propagation only carries out necessary calculations on certain paths as explained in Section 4. The results also show that as networks become large, the savings of path propagation compared with GP are significant.

## 6 Remarks and Conclusion

Throughout the paper, we have assumed that  $v(\mathbf{E})$  is contained by a clique. If  $v(\mathbf{E})$  is not contained by any clique in a junction tree, for example, consider the evidence  $\mathbf{E}$  denoting  $a = 1$ ,  $g = 0$ , suppose  $v(\mathbf{E}) = \{a, g\}$  is not contained in any clique in a junction tree, one can always decompose  $\mathbf{E}$  as  $\mathbf{E}_1$  denoting  $a = 1$  and  $\mathbf{E}_2$  denoting  $g = 0$  such that  $v(\mathbf{E}_1) = \{a\}$  and  $v(\mathbf{E}_2) = \{g\}$  must be contained by a clique because they are singleton sets after decomposition.

We only discussed two query patterns in Section 4. In reality, pattern 1 and 2 can interweave with each other in a query session. This interweaving case can be similarly solved as pattern 1 and 2. We have also based our discussion of query pattern 2 on the assumption that evidences  $\mathbf{E}_1$ ,  $\mathbf{E}_2$ , ..., and  $\mathbf{E}_n$  are compatible. Due to the length limit of the paper, we postpone the discussion of the interweaving case and contradicting evidence case in an extended version of this paper.

The implementation of path propagation involves as a major operation finding a path in a junction tree connecting the clique containing the variable of interest and the clique containing the evidence. Finding a path in a tree can be effectively and efficiently implemented. Since the proposed method for answering queries only involves a few paths in a junction tree based on the queries imposed instead of the whole junction tree, it takes less time and resources to compute the answer for a query, especially for large and complex BNs in which the GP method may fail to operate effectively and efficiently.

## References

- [1] O. Bangs and P. Wuillemin. Top-down construction and repetitive structures representation in bayesian networks. In *Proceedings of the Thirteenth International FLAIRS Conference, Florida, USA, 2000.*, pages 282–286, 2000.
- [2] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, October 1996.
- [3] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, 4: 269–282, 1990.
- [4] S.L. Lauritzen and D.J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50:157–244, 1988.
- [5] Vasilica Lepar and Prakash P. Shenoy. A comparison of Lauritzen-Spiegelhalter, Hugin, and Shenoy-Shafer architectures for computing marginals of probability distributions. In Gregory F. Cooper and Serafin Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 328–337, San Francisco, July 24–26 1998. Morgan Kaufmann.
- [6] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Francisco, California, 1988.
- [7] Y. Xiang. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge, 2002.

# Problem-Solving Knowledge Mining from Users' Actions in an Intelligent Tutoring System

Roger Nkambou<sup>1</sup>, Engelbert Mephu Nguifo<sup>2</sup>, Olivier Couturier<sup>2</sup>,  
and Philippe Fournier-Viger<sup>1</sup>

<sup>1</sup> Université du Québec à Montréal (Canada)

<sup>2</sup> CRIL-CNRS, IUT de Lens (France)

nkambou.roger@uqam.ca, mephu@cril.univ-artois.fr

**Abstract.** In an intelligent tutoring system (ITS), the domain expert should provide relevant domain knowledge to the tutor so that it will be able to guide the learner during problem solving. However, in several domains, this knowledge is not predetermined and should be captured or learned from expert users as well as intermediate and novice users. Our hypothesis is that, knowledge discovery (KD) techniques can help to build this domain intelligence in ITS. This paper proposes a framework to capture problem-solving knowledge using a promising approach of data and knowledge discovery based on a combination of sequential pattern mining and association rules discovery techniques. The framework has been implemented and is used to discover new meta knowledge and rules in a given domain which then extend domain knowledge and serve as problem space allowing the intelligent tutoring system to guide learners in problem-solving situations. Preliminary experiments have been conducted using the framework as an alternative to a path-planning problem solver in CanadarmTutor.

## 1 Introduction

In an intelligent tutoring system (ITS), the domain expert should provide relevant domain knowledge to the tutor so that it will be able to guide the learner during problem solving. However, in several domains, this knowledge is not predetermined and should be captured or learned from expert users as well as intermediate and novice users. Our hypothesis is that, knowledge discovery (KD) techniques can help to build this domain intelligence in ITS.

This paper proposes an approach to support new domain knowledge discovery in domain where it is difficult to set up a clear problem space or task models. In such a domain, we need to capture new procedures (correct or incorrect), new problem spaces and new problem-solving strategies from users' actions. Cognitive task analysis that aims at producing effective problem space or task model (to support model and knowledge tracing, coaching, errors detection and plan recognition) is a very time consuming process [8]. How can we build this complex structure by learning from users' interactions with an ITS ?

The approach presented in this paper is based on a combination of sequential pattern recognition and association rule discovery. We show how the proposed approach

is used to discover new knowledge in a given domain, which then extends domain knowledge and serves as a problem space allowing the intelligent tutor to track learners' actions and give relevant hints when needed.

The paper is organized as follows. First, we will present the context of this research work by stating the need of KD to enhance tutoring agent knowledge. Then we will describe the tutoring context and show how data can be transformed for KD. We will also briefly describe algorithms that take this context as input, to extract significant sequences of patterns and relationships between them, which will constitute relevant partial or complete plans reusable in a given problem-solving activity to track student cognitive behavior. Finally, we show how this knowledge is used by a tutoring system (CanadarmTutor) aimed at training astronauts during procedural tasks on the ISS (International Space Station) using a robot manipulator called CanadarmII.

## 2 Problem Statement and Related Works

Educational data mining is becoming a very important area in the Artificial Intelligence in Education community [1]. Several techniques are used to extract relevant data, information or knowledge mainly from databases and log files of learning sessions. However, most work focuses on learner or group classification, clustering or sorting [2, 3]. Very few studies address procedural knowledge learning and none attempts to find and learn relations between actions, sequences of actions, and patterns among them, which may provide useful information regarding the procedure.

Kay *et al.* [4] describe student group interaction data mining that seeks to identify significant sequences of activity. Their goal is to flag interaction sequences which indicate problems and successes, so that tutors can help students recognize problematic situations in the early stages of the learning sessions. Their goal is not related to learning procedures nor does it aim to find links between significant interaction sequences or patterns.

Very little AIED research investigated ITS automatic procedural knowledge learning [5, 6, 7]. Yet, such a capability could facilitate the development of problem spaces (task model, procedural knowledge, etc.) and reduce the need for domain experts. For example, [6] attempted to induce simple production rules using a single example and the analogy mechanism in ACT-R; [7] looks up a set of marked examples, trying to generalize them and generate production rules. None of them have explored sequential patterns and rules discovery, which can help determine problem-solving steps and rules.

Creating cognitive tutors usually rests on the implicit assumption that one should predefine a task model describing correct and incorrect solution paths. Similarly, CTAT (Cognitive Tutor Authoring Tool) [8] offers a set of tools that allows ITS designers to specify the behavioural graph (BG) of a task, presenting correct and buggy paths. BGs (sometimes transformed into production rules) are used to track student actions. The behaviour recorder can automate the translation of user actions into a BG. This concept was improved by the BND (Bootstrapping Novice Data) approach proposed by McLaren *et al.* [5]. BND records the actions of many students in a log file which is then used to create a common BG that can be improved by designers. Instead of having authors build problem-solving expertise from scratch, tap into only their own experience or incorporate student data manually as in traditional ITS

development, this tool semi-automatically leverages the empirical data of actual problem-solving activities. However, the BND approach is devoid of data mining and learning, reducing the approach to a simple way of storing or integrating raw user solutions into a structure, as in [6] and [7]. In fact, student data are incorporated into the BG regardless of possible links between problem steps or actions. This is very limiting because the system does not try to extract useful knowledge from those solutions, which could enrich the problem space.

Contrary to these approaches, we are proposing a solution to create a more general, flexible and powerful (albeit sometimes partial) BG-like structure by inferring association rules between actions or action sequences. In fact, domain users (both expert and novice) can provide primitive action sequences required to achieve typical tasks in the application domain. These sequences (good and buggy) may then be used to teach procedural knowledge associated with the task, thereby continually enhancing the system's intelligence.

### 3 Modelling Procedural Knowledge in CanadarmTutor

One of the main goals of an intelligent tutoring system is to actively provide relevant feedback to the student in problem-solving situations [9]. This kind of support becomes very difficult when an explicit representation of the training task is not available. This is the case in the ISS environment where the problem space associated with a given task consists of an infinite number of paths. Moreover, there is a need to generate new tasks on the fly without any cognitive structure. *Roman Tutor* brings a solution to these issues by using FADPRM, a path planner, as main resource for the tutoring feedback.

FADPRM [10] is a flexible and efficient approach for robot path planning in constrained environments. In addition to the obstacles that the robot must avoid, our approach holds account of desired and non-desired (or dangerous) zones. This will make it possible to take into account the disposition of cameras on the station. Thus, our planner will try to bring the robot in zones offering the best possible visibility of the progression while trying to avoid zones with reduced visibility.

FADPRM allows us to put in the environment different zones with arbitrary geometrical forms. A degree of desirability  $dd$ , a real in  $[0\ 1]$  is assigned to each zone. The  $dd$  of a desired zone is then near 1, and the more it approaches 1, the more the zone is desired; the same for a non-desired zone where the  $dd$  is in  $[0\ 0.5]$ . On the international Space Station, the number, the form and the placement of zones reflect the disposition of cameras on the station. A zone covering the field of vision of a camera will be assigned a high  $dd$  (near 1) and will take a shape which resembles that of a cone; whereas a zone that is not visible by any camera from those present on the station will be considered as a non-desired zone with a  $dd$  near to 0 and will take an arbitrary polygonal shape.

The ISS environment is then preprocessed into a roadmap of collision-free robot motions in regions with highest desirability degree. More precisely, the roadmap is a graph such that every node  $n$  is labeled with its corresponding robot configuration  $n.q$  and its degree of desirability  $n.dd$ , which is the average of  $dds$  of zones overlapping with  $n.q$ . An edge  $(n,n')$  connecting two nodes is also assigned a  $dd$  equal to the

average of  $dd$  of configurations in the path-segment  $(n.q,n'.q)$ . The  $dd$  of a path (i.e., a sequence of nodes) is an average of  $dd$  of its edges.

Following probabilistic roadmap methods (PRM) [11], we build the roadmap by picking robot configurations probabilistically, with a probability that is biased by the density of obstacles. A path is then a sequence of collision free edges in the roadmap, connecting the initial and goal configurations.

Following the Anytime Dynamic A\* (AD\*) approach [12], to get new paths when the conditions defining safe zones have dynamically changed, we can quickly re-plan by exploiting the previous roadmap. Moreover, paths are computed through incremental improvements so that the planner can be called at anytime to provide a collision-free path and the more time it is given, the better the path optimizes moves through desirable zones. Therefore, our planner is a combination of the traditional PRM approach [11] and AD\* [12] and it is flexible in that it takes into account zones with degrees of desirability. This explains why we called it Flexible Anytime Dynamic PRM (FADPRM).

We implemented FADPRM as an extension to the Motion Planning Kit (MPK)[11] by changing the definition of PRM to include zones with degrees of desirability and changing the algorithm for searching the PRM with FADPRM. The calculation of a configuration's  $dd$  and a path's  $dd$  is a straightforward extension of collision checking for configurations and path segments.

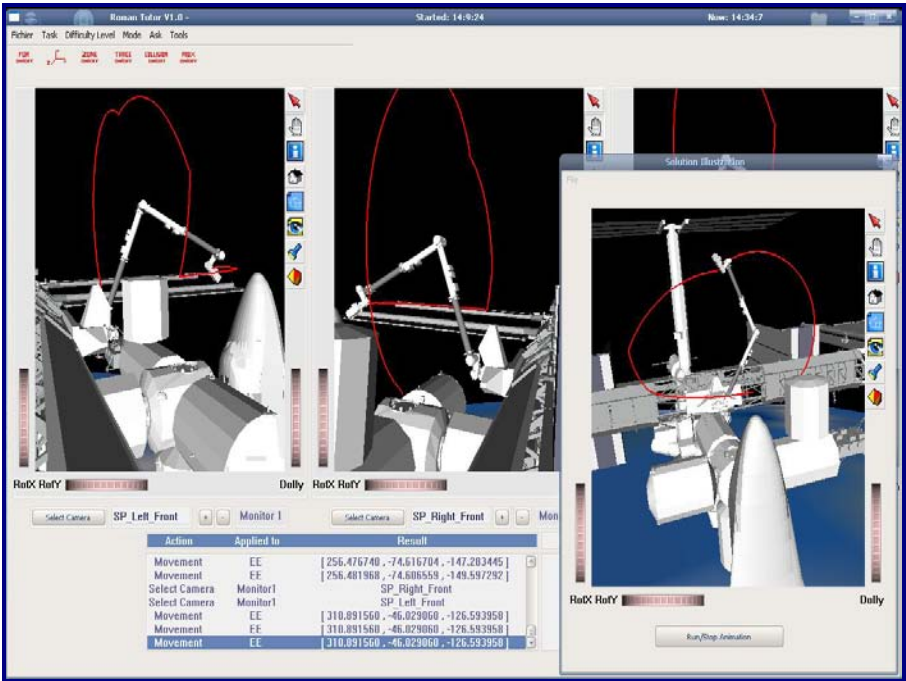


Fig. 1. Path planning and task demonstration using FADPRM

FADPRM computes a collision and soft constraints (camera views, etc.) free path that serves as expert solution for the tutor. FADPRM is also called when the tutor needs to validate student actions, to demonstrate a given task or to suggest a solution path. However, the resulting path is sometimes too much complex to be followed by a human user and far from the procedure that a human would execute in a real-world situation. In figure 1, FADPRM generates a path that the user should follow and can demonstrate it in another window.

A good tutor in procedural tasks should fulfill at least the following important properties: 1) guide the user through expert users' solution; 2) and recognize the student profile (novice, intermediate or expert) to offer tailored help.

Tutoring services based on FADPRM fails to satisfy these properties. We believe that a way to solve this problem is to base coaching on knowledge that comes from users themselves. In this way, the system can 1) capture data from the system usage by users of all possible profiles and 2) learn rules and constraints that can contribute to a knowledge base to support adapted tutoring services. Our hypothesis is that, tutoring services based on such a knowledge base will guarantee high quality assistance to the learners.

The next sections of this paper present a way to implement this solution.

## 4 Problem-Solving Data Representation

In cognitive tutors, problem-solving knowledge is represented as procedures each corresponding to a possible path to a successful or unsuccessful solution to the problem. A procedure (or a plan) is a sequence of atomic and non-atomic actions. Non-atomic actions are actions containing at least two atomic actions. Actions are events that occur at a given time. Table 1 shows an example dataset of 8 successful plans where each entry corresponds to a plan's events. From this dataset, it is possible to easily compute frequent sequences of actions using a minimal support (*minsup*) defined by the user. A sequence is said to be frequent if it occurs more than *minsup* times.

**Table 1.** A data set of 8 successful plans

PlanID	Sequences of actions
P1	1 2 25 46 48 {9 10 11 31}
P2	1 25 46 54 79 {10 11 25 27}
P3	1 2 3 {9 10 11 31} 48
P4	2 3 25 46 11 {14 15 16 48} 74
P5	2 25 46 47 48 49 {8 9 10}
P6	1 2 3 4 5 6 7
P7	25 26 27 28 30 {32 33 34 35 36}
P8	46 54 76 {10 27} {48 74}

From the frequent sequences set, the next step consists in finding rules that connect them using a simple algorithm that considers sub-sequences of each sequence and derives a relationship between them given their number of occurrences in the dataset.



## 5 The Proposed Framework

The system that we propose goes through different stages or processes to learn rules. At each stage, we adapt and integrate specific algorithms. The main scheme is as follow (process in bold):

*Log files containing users plans*  $\rightarrow$  **Automatic coding of data**  $\rightarrow$  *Formatted Binary-File*  $\rightarrow$  **Sequential Patterns Finding (PrefixSpan)**  $\rightarrow$  *Frequent patterns*  $\rightarrow$  **Building of the Meta-Context**  $\rightarrow$  *Meta-Context*  $\rightarrow$  **Association Rules Finding (IGB)**  $\rightarrow$  *New procedural task knowledge (PTK)*  $\rightarrow$  **Integration within the Tutoring System.**

### 5.1 Sequential Patterns Mining

The problem of mining sequential patterns was originally proposed by Agrawal and Srikant [11]. Let  $I = \{i_1, i_2, \dots, i_n\}$  be a set of items or actions. We call a subset  $X \subseteq I$  an *itemset* or an *actionset* and we call  $|X|$  the *size* of  $X$ . A *sequence*  $s = (s_1, s_2, \dots, s_m)$  is an ordered list of actionsets, where  $s_i \subseteq I, i \in \{1, \dots, m\}$ . The size,  $m$ , of a sequence is the number of actionsets in the sequence, i.e.  $|s|$ . The length of a sequence  $s = (s_1, s_2, \dots, s_m)$  is defined as :  $l = \sum |s_i|$ , for  $i = 1$  to  $m$ .

A sequence with length  $l$  is called an  $l$ -sequence. A sequence  $s_a = (a_1, a_2, \dots, a_n)$  is contained in another sequence  $s_b = (b_1, b_2, \dots, b_m)$  if there exists integers  $1 \leq i_1 < i_2 < \dots < i_n \leq m$  such that  $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_n \subseteq b_{i_n}$ . If sequence  $s_a$  is contained in sequence  $s_b$ , then we call  $s_a$  a *subsequence* of  $s_b$  and  $s_b$  a *supersequence* of  $s_a$ .

The *relative support* is defined as the percentage of sequences  $s \in D$  that contains  $s_a$ . The support of  $s_a$  in  $D$  is denoted by  $\text{sup}D(s_a)$ .

Given a support threshold  $\text{minsup}$ , a sequence  $s_a$  is called a *frequent sequential pattern* on  $D$  if  $\text{sup}D(s_a) \geq \text{minsup}$ . The problem of mining sequential patterns is to find all frequent sequential patterns for a database  $D$ , given a support threshold  $\text{minsup}$ .

Table 1 shows the dataset consisting of tuples in its sequence representation. Consider the sequence of plan 2; the size of this sequence is 6, and the length of this sequence is 9. Suppose we want to find the support of the sequence  $s_a = (I \{9 \ 3I\})$ . From Table 1, we know that  $s_a$  is a subsequence of the sequences for plan 1 and plan 3 but is not a subsequence of the sequence for plan 2. Hence, the support of  $s_a$  is 2 (out of a possible 8), or 0.25. If the user-defined minimum support value is less than 0.25, then  $s_a$  is deemed frequent.

A subsequence or pattern,  $P$ , is *closed* if there exists no superset of  $P$  with the same support in the database. A closed pattern induces an equivalence class of pattern sharing the same closure. The *minimal generators* and the unique *closed pattern* of an equivalence class of actionsets share a common set of plans. The minimal generators are the minimal ones among the equivalent actionsets, while the closed pattern is the maximum one. The closed pattern is unique.

### 5.2 Finding Frequent Sequential Patterns Using PrefixSpan

Many algorithms have been proposed to efficiently mine sequential patterns or other time-related data [13, 14, 15, 16]. We choose here the PrefixSpan approach [15] as it is

one of the most promising ones for mining large sequence databases having numerous patterns and/or long patterns, and also because it can be extended to mine sequential patterns with user-specified constraints. PrefixSpan is a projection-based, sequential pattern-growth approach that recursively projects a sequence database into a set of smaller projected databases. Sequential patterns are grown in each projected database by exploring only locally frequent fragments. Table 2 shows examples of sequential patterns extracted by PrefixSpan from data in table 1 using a minimum support equals 25%.

Links between sequential patterns can lead to the tutor goal. Thus PrefixSpan can find long frequent sequence patterns, and those patterns will be linked by generating associations among them. In our case, we are interested by minimal and non redundant association rules, also called generic bases.

**Table 2.** Examples of sequential patterns extracted by PrefixSpan with their associated labels

Sequential patterns	Sequence patterns' labels
1 25 46 48	S1
1 25 46 {10 11}	S2
1 {9 10 31}	S6
1 {9 11 31}	S7
1 {9 10 11 31}	S8
1 46 {10 11}	S13

Among previous studies on mining of generic bases, we choose IGB [17] as it efficiently extracts more compact generic bases without information loss, i.e. all association rules can be derived from these generic bases with their exact support.

### 5.3 Extracting Generic Rules Between Patterns Using IGB

IGB [17] is a new informative generic basis. It has a valid and complete axiomatic system allowing the derivation of all the association rules. Rules of IGB are correlations between minimal premise and maximal conclusion (in term of items number). Indeed, it was shown that this kind of rules is the most general (i.e., conveying the maximum of information). The premise of some generic rules of IGB can be empty such that they are two types of generic rules: (1). *factual rules* having an empty premise; and (2). *implicative rules* having a non empty premise.

IGB basis is generated by a dedicated algorithm which takes as input the meta-context of initial plans, and two thresholds which are the minimum support, *minsup* (already defined in PrefixSpan), and the minimum confidence, *minconf*. The meta-context of initial plans (see example in Table 3) is the set of plans redefined with the frequent sequential patterns obtained with PrefixSpan.

IGB algorithm checks for each non empty closed pattern,  $P$ , if its support is greater or equal to *minconf*. If it is the case, then the generic rule  $\emptyset \rightarrow P$  is added to IGB base. Else, it iterates on all frequent closed actionsets  $P_0$  subsumed by  $P$ . For those having support at least equal to  $\text{support}(P)/\text{minconf}$ , the algorithm iterates on the list of minimal generators associated to  $P_0$ . During this iteration, we look for the smallest minimal generator  $g_s$ , such that there does not exist a generator  $g_0$  subsumed by  $g_s$  which is already inserted in the list  $L$  of smallest premises. Then, IGB algorithm iterates on all elements of the list  $L$  in order to generate rules of IGB which have the following form:  $g_s \rightarrow (P - g_s)$ .

**Table 3.** Part of the crisp meta-context of frequent sequences built from dataset in table1

PlanID	Frequent sequential patterns
P1	S1 S2 S4 S5 S6 S7 S8 S9 S10 S95 S97 S98 S113 S116 S118
P2	S1 S5 S6 S7 S9 S98
P3	S1 S2 S3 S4 S5 S6 S8 S10 S95 S97 S98 S113 S116 S118
P4	S2 S3 S6 S7 S9 S10
P5	S2 S4 S5 S7 S9 S10 S95
P6	S1 S2 S3
P7	S7
P8	S5 S9 S10

By dividing the sub-sequence occurrence by the plans’ occurrence, we obtain the relative support associated to the sub-sequence. Let consider a *minsup* of 2 (25%), meaning that a valid sequence should occur in at least 2 input-plans, we can obtain the meta-context which part is shown in table 3. Each sub-sequence can appear in a plan with a certain confidence which is its relative support (in table 3, we consider a crisp context where dichotomic values (0 or 1) are assigned when a subsequence appears or not in a plan). Using this meta-context as input, IGB computes a set of generic meta-rules, part of which is shown in table 4. These meta-rules combined with frequent sequential patterns will constitute the knowledge that will be used by the tutor to guide students and domain users to explore and learn the procedural task.

**Table 4.** Examples of generic meta-rules extracted by IGB

Meta-rules	Support	Confidence	Expanded meta-rules
S10 ==> S9	4	0.8	...
S9 ==> S7	4	0.8	1 {10 31} ==> 1 {9 11 31}
S9 ==> S5	4	0.8	...
S5 ==> S10	4	0.8	...

## 6 How the Learned Knowledge Base Is Used for Tutoring Services?

As said before, tutoring systems should provide useful tutoring services to assist the learner. These services include coaching, assisting, guiding, helping or tracking the student during problem-solving situations. To offer these services, a tutoring system needs some knowledge related to the context. The knowledge base namely procedural task knowledge (PTK) obtained from the previous knowledge mining process serves to that end. The next paragraphs present some examples of services that can be supported.

**Assisting the User to Explore Possible Solutions of a Given Problem.** Domain expert users can explore, validate or annotate the PTK. The validation can consist in removing all meta-rules with a low confidence, meaning that those rules can not significantly contribute to help the student. Annotation consists in connecting some useful information to meta-rules lattice depicting semantic steps of the problem as well as

hints or skills associated to a given step. A meta-rule lattice annotated in this way is equivalent to [8]'s BN or Sherlock's effective problem space (EPS), except that EPS and BN are explicitly built from scratch by domain experts.

For student users, exploring PTK will help them learn about possible ways of solving problem. They can be assisted in this exploration using an interactive dialog with the system which can prompt them on their goals and helps them go through the rules in order to achieve these goals. This kind of service can be used when the tutoring system wants to prepare students before involving them in real problem-solving situation.

**Tracking the Learner Actions to Recognize the Plan S/He is Following.** Plan recognition is very important in tutoring systems. PTK is a great resource to this process. Each student's action can be tracked by searching the space defined by meta-rules lattice so that we can see the path being followed. For this service, partitioning the space in terms of equivalent classes corresponding to maximal sequences as proposed in [14] can make plan recognition (and exploration) easier. In fact, when it is recognized the current plan is in a class, all other classes are pruned so that the exploration will continue only in a single class.

**Guiding Learners.** When solving a problem, an ITS should be able to help the student. A classic situation is when the student asks the tutor what to do next from the actual state. PTK can help the tutoring agent to produce the next most probable actions that the student should execute and prompt him on that, taking into account uncertainty related to rules' confidence. An example of a dialog can be as follow:

....

*Student : What should I do now ?*

*Tutor : Oh! I don't quite know but I think you should try action B.*

*Student : Why ?*

*Tutor : Well, in 75% of the cases, people who tried that action achieved the final goal !*

*Student : Ok! Are there any other possibilities?*

...

## 7 Results and Discussion

We have set up two scenarios consisting each in moving the load to one of the two cubes (figure 2a). A total of 15 users (a mix of novices, intermediates and experts) have been invited to execute these scenarios using the CanadarmII robot simulator. A total of 119 primitive actions have been identified. Figure 2b shows part of an example log file from a user's execution of the first scenario. We obtain a database with 45 entries each corresponding to a given usage of the system. A value indicating the failure or success of the plan has been added at the end of each entry.

The framework presented in section 5 was applied. A unique number was assigned to each action. After coding each entry of the traces database using PrefixSpan data representation, we obtained a binary file containing plans' data for the two scenarios. This file was sent as input to the rest of the process.

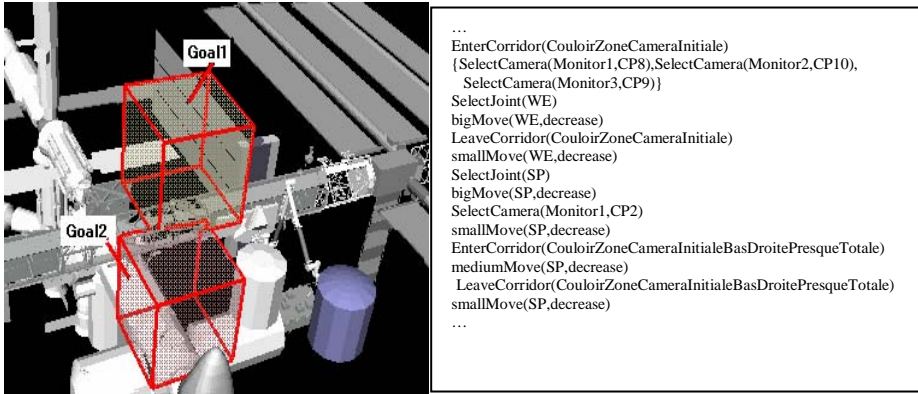


Fig. 2. (a) Environment setup for the two experimental scenarios

(b) An entry of the plans' database

**The Results.** After executing PrefixSpan, the first stage of the experiment consisting in finding sequential patterns from the input data, we obtained a total of 76 significant patterns (with a *support* greater than .5). At the second stage, we created a binary context where each row represents a plan data and each column stands for a set of patterns. The goal at this stage was to mine association rules between sequential patterns. Using IGB approach, we obtained a PTK of 37 significant meta-rules. These rules were then coded and integrated in a new version of CanadarmTutor that uses this knowledge base to support tutoring services. An empirical test with this version has been conducted with the same users of the system's version relying on the FADPRM. They have been asked to execute the two scenarios. We found that, the system behavior in terms of guiding the user was significantly improved compared to the behavior observed in the version relying on the path planner. The system can now recommend good and easy-to-follow actions sequences. The system can also recognize users' plans and anticipate failures or successes, thus proactively help them at the earlier stage. Using the learned knowledge base, the system can also infer user profiles by detecting (analyzing) the path they follow. The PTK produced by our framework is sometimes too large and contains non useful rules. However, this is not harmful for the tutor behavior but it may slow the performance as the system need to go through this huge knowledge base each time the user executes an action. We are now working to improve the quality of the PTK. We are also looking for a way of managing unsuccessful plans data. In fact in the actual version of the implemented framework, we do not consider plans that fail. We should find a way to integrate these data. We believe that this integration may lead to a more powerful behavior of the tutoring agent in the sense that it can easily identify sequence patterns that lead to failure or success, hence better guiding the learner.

## 8 Conclusion

In this paper, we proposed a KD framework that combines sequential pattern mining and association rules discovery techniques. We showed how the proposed framework can contribute to enhance an intelligent tutoring system's knowledge in procedural

domain. We used the framework to build a meta-knowledge base of plans from users' traces in CanadarmTutor. The resulting knowledge base enables CanadarmTutor to better help the learner. For future works, we plan to find some ways of filtering the resulting meta-rules and integrating unsuccessful paths. We will also carry out further tests to clearly measure the benefit of the approach in terms of tutoring assistance services.

## Acknowledgment

We would like to thank Kalhed Belghith and Froduald Kabanza for their contributions to the FADPRM design and implementation. We also thank Sadok Ben Yahia and Ghada Gasmi for fruitful discussions on this work.

## References

1. Heiner, C., Baker, R. and Yacef, K. (2006). *Proceedings of the Educational Data Mining Workshop*. ITS'2006.
2. Graf, S. and Bekele, R. (2006). "Forming Heterogeneous Groups for Intelligent Collaborative Learning Systems with Ant Colony Optimization". *Intelligent Tutoring Systems 2006*: 217-226. Springer-Verlag.
3. Amershi, S. and Conati, C. (2006). "Automatic Recognition of Learner Groups in Exploratory Learning Environments". *Intelligent Tutoring Systems 2006*: 463-472. Springer-Verlag.
4. Kay, J., Maisonneuve, N., Yacef, K., Zaïane, O. (2006). "Mining Patterns of Events in Students' Teamwork Data". *Proceedings of the Workshop on Educational Data Mining Workshop*. ITS'2006, pp. 45-52.
5. McLaren, B. M. et al. (2004). "Bootstrapping Novice Data: Semi-Automated Tutor Authoring Using Student Log Files". *Proceedings of the Workshop on Analyzing Student-Tutor Logs*. ITS'2004.
6. Blessing, S.B. (2003). "A Programming by Demonstration Authoring Tool for Model-Tracing Tutors". In, *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*, pp. 93-119. Kluwer Academic Publishers
7. Jarivs, M., Nuzzo-Jones, G. & Heffernan, N. T. (2004). "Applying Machine Learning Techniques to Rule Generation in Intelligent Tutoring Systems". *Intelligent Tutoring Systems 2006*: 541-553. Springer.
8. Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. (2006). "The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains". *Intelligent Tutoring Systems 2006*: 61-70. Springer.
9. VanLehn, K. (2003). "The advantages of Explicitly Representing Problem Spaces". *User Modeling*, Springer Verlag LNAI 2702:3.
10. Kabanza, F., Nkambou, R. and Belghith, K. (2005). "Path-Planning for Autonomous Training on Robot Manipulators in Space". *IJCAI 2005*: 1729-1731.
11. Sanchez, G., Latombe, J.C. (2001). "A single-query bi-directional probabilistic roadmap planner with lazy collision checking". *Int. Symposium on Robotics Research (ISRR'01)*. Springer Tracts in Advanced Robotics, Springer, 403-417.

12. Likhachev, M., Ferguson, D., Stentz, A., Thrun, S. (2005). "Anytime Dynamic A\*: An Anytime Replanning Algorithm". *Proceedings of International Conference on Automated Planning and Scheduling*.
13. R. Agrawal and R. Srikant. Mining Sequential Patterns. *Proceedings of the 1995 Int. Conference on Data Engineering*, pp. 3-14, 1995
14. Zaki, M.J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning Journal*, 42(1-2): 31-60.
15. J. Pei, J. Han et al (2004). Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE Transaction on Knowledge and Data Engineering*, 16(10), oct. 2004.
16. F. Masseglia, F. Cathala, and P. Poncelet (1998), "The PSP Approach for Mining Sequential Patterns". *Proceedings European Symp. Principle of Data Mining and Knowledge Discovery (PKDD '98)*, pp. 176-184.
17. Gasmi G., Ben Yahia S., Mephu Nguifo E., Slimani Y. (2005). "A new informative generic base of association rules", *The Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-05)*, LNCS, Springer Verlag, Hanoi, Vietnam.

# Incremental Neighborhood Graphs Construction for Multidimensional Databases Indexing

Hakim Hacid<sup>1</sup> and Tetsuya Yoshida<sup>2</sup>

<sup>1</sup> Lyon 2 University

ERIC Laboratory- 5, avenue Pierre Mendès-France  
69676 Bron cedex - France

<sup>2</sup> Hokkaido University

Grad. School of Information Science and Technology,  
N-14 W-9, Sapporo 060-0814, Japan

hhacid@eric.univ-lyon2.fr, yoshida@meme.hokudai.ac.jp

**Abstract.** The point location (neighborhood search) is a significant problem in several fields like databases and data mining. Neighborhood graphs are interesting representations of this problem in a multidimensional space. However, several problems related to neighborhood graphs are under research and require detailed work to solve them. These problems are mainly related to their high construction costs and to their updating difficulties. In this article, we deal with the point location problem by considering neighborhood graphs optimization. We propose and compare two strategies able to quickly build and update these structures.

## 1 Introduction

The point location problem<sup>1</sup> is a key question in the automatic multidimensional data processing. Indeed, lots of algorithms and techniques in different domains are based on this concept. We can quote for example *k-means*[9], Kohonen maps [12], and *k-NN* in data mining or the majority of the databases indexing techniques [3]. This gives evidence to the interest and the importance of the neighborhood search concept.

The point location problem can be stated as follows: having a set of data  $\mathbf{V}$  with  $n$  items in a multidimensional space  $\mathcal{R}^d$ , the problem is then to find a way to pre-process the data so that if we have a new query item  $q$ , we'll be able to find its neighbors within as short as possible time. Among the possible representations of this problem in a multidimensional space, neighborhood graphs are very interesting and very popular. Their popularity is due to the fact that the neighborhood is determined by coherent functions which reflect, in some point of view, the mechanism of the human intuition for the neighborhood determination. We will use the following notations throughout this paper.

Let  $\mathbf{V}$  be a set of points (vertexes) in a multidimensional space  $\mathcal{R}^d$ . A graph  $G(\mathbf{V}, \mathbf{E})$  is composed of a set of vertexes  $\mathbf{V}$  and a set of edges  $\mathbf{E}$ . Then, for

---

<sup>1</sup> The "point location problem" concept is equivalent to "neighborhood search" in the rest of this article.



any graph we can associate a binary relation upon  $\mathbf{V}$ , in which two points  $(v_1, v_2) \in \mathbf{V}^2$  are in binary relation if and only if the couple  $(v_1, v_2) \in E$ . In an other manner,  $(v_1, v_2)$  are in binary relation  $(R)$  if and only if they are directly connected in the graph  $G$ . From that, the neighborhood  $N(v_1)$  of a vertex  $v \in \mathbf{V}$  in the graph  $G$  can be considered as a set of vertexes which are directly connected to  $v$ .

In this paper, we deal with the point location in a multidimensional space, and this, in order to find an efficient way for optimizing the construction task of the neighborhood graphs. We propose two strategies based on an intelligent manner for locally updating the neighborhood of each point. The first construction strategy, a *forward strategy*, starts from an empty set of points and adds one point (as well as its corresponding connexions) at each iteration to the "graph set". The second strategy, *backward strategy*, as for it starts from an already built structure (We consider in our case a Delaunay triangulation [10]) and drops the connexions one by one until obtaining the desired neighborhood graph. This second strategy exploits some inclusion relations between the considered structures.

The rest of this article is organized as follows: Section 2 presents a state of art on the point location problem, on neighborhood graphs as well as the problematic. Our proposition, based on the search of an optimal hyper sphere, is presented in Section 3. Next, we present the performed experiments and the obtained results. Finally, we conclude and give some future works in Section 5.

## 2 State of the Art

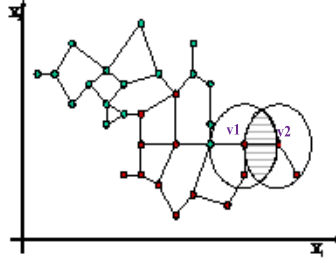
As pointed in the introduction, the objective of the point location is to find a way to pre-process the data so that if we have a new query item, we'll be able to find its neighbors within as short as possible time. One possible way to represent such data structure is a neighborhood graph. Neighborhood graphs, or proximity graphs, are geometrical structures which use the concept of neighborhood to determine the closest points to another. For that, they are based on the distance measures [14].

Several possibilities were proposed for building neighborhood graphs. Among them we can quote the Delaunay triangulation [10], the relative neighborhood graphs [13], the Gabriel graph [2]. For illustration, we describe hereafter the relative neighborhood graph ( $RNG$ ).

In a relative neighborhood graph  $G_{RNG}(\mathbf{V}, \mathbf{E})$ , two points  $(v_1, v_2) \in \mathbf{V}^2$  are neighbors if they check the relative neighborhood property defined as follows: Let  $\mathcal{H}(v_1, v_2)$  be the hyper-sphere of radius  $d(v_1, v_2)$  and centered on  $v_1$ , and let  $\mathcal{H}(v_2, v_1)$  be the hyper-sphere of radius  $d(v_2, v_1)$  and centered on  $v_2$ . Note that  $d(v_1, v_2)$  and  $d(v_2, v_1)$  are the dissimilarity measures between the two points  $v_1$  and  $v_2$ , and  $d(v_1, v_2) = d(v_2, v_1)$ .

So,  $v_1$  and  $v_2$  are neighbors if and only if the lune  $\mathcal{A}(v_1, v_2)$  formed by the intersection of the two hyper-spheres  $\mathcal{H}(v_1, v_2)$  and  $\mathcal{H}(v_2, v_1)$  is empty [13]. Formally:

$$\mathcal{A}(v_1, v_2) = \mathcal{H}(v_1, v_2) \cap \mathcal{H}(v_2, v_1) \quad \text{Then } (v_1, v_2) \in \mathbf{E} \text{ iff } \mathcal{A}(v_1, v_2) \cap \mathbf{V} = \emptyset$$



**Fig. 1.** An illustration of a relative neighborhood graph in a bi-dimensional space. The relative neighborhood property is the empty of the intersection of two hyper-spheres.

Fig. 1 illustrates the relative neighborhood graph in a bi-dimensional space.

Several algorithms for neighborhood graphs construction were proposed. Since we are mainly interested in RNG in this paper, the algorithms which we quote briefly hereafter relate to the construction of RNG. Other algorithms for Delaunay triangulation are discussed in detail in [15].

One of the common approaches to the various neighborhood graphs construction algorithms is the use of the refinement techniques. In this approach, the graph is built by steps. Each graph is built starting from the previous graph, containing all connections, by eliminating some edges which do not check the neighborhood property of the graph to be built. Pruning (edges elimination) is generally done by taking into account the construction function of the graph or through geometrical properties.

The construction principle of the neighborhood graphs consists in seeking for each point if the other points in the space are in its proximity. The cost of this operation is with a complexity of  $O(n^3)$  ( $n$  is the number of points in the space). Toussaint [14] proposed an algorithm of complexity  $O(n^2)$ . He deduces the RNG starting from a Delaunay triangulation [10]. Using the Octant neighbors, Katajainen [7] also proposed an algorithm with the same complexity. Smith [11] proposed an algorithm of complexity  $O(n^{23/12})$  which is less significant than the standard complexity ( $O(n^3)$ ).

These last algorithms have a low complexity comparing to the standard one ( $< O(n^3)$ ), so they are focused on a fast way for building the graph. Unfortunately, these algorithms do not support the updating task. Indeed, if one want to insert a new point in the graph, the algorithm have to rebuild all the graph. This can be interesting in a learning task with data samples, generally with few items, to generate rules. For an indexing task, these algorithms are not suitable.

### 3 Optimizations on Neighborhood Graphs

Lots of efforts have been carried out in the domain of computational geometry to optimize topological models. The objective is mainly to reduce the computational complexity of these structures. We aim to propose efficient strategies able to

build incrementally and quickly these structures without rebuilding the whole structure at each modification in the graph. In what follows, we introduce the core method<sup>2</sup> of all the incremental construction strategies discussed in this article.

### 3.1 Local Insertion in a Neighborhood Graph

The optimization of the neighborhood graphs construction passes by the location of the inserted/removed point as well as all the affected edges (update propagation). To achieve this, we proceed in two main stages : initially, we determine an optimal space area (a hyper sphere in our case) which can contain a maximum number of potentially closest points to the query point (the point to locate in the multidimensional space). The second stage is performed in order to calculate the updating propagation and effectively applying the modifications. Thus, this last stage causes the effective updating of the neighborhood relations between the concerned points.

The main stage in this method is the search area determination. This can be considered as a question of determining a hyper sphere which maximizes the chance of containing the neighbors of the query item while minimizing the total number of items that it contains.

With regard to the first step (determine the radius of an hyper sphere  $SR$  which maximizes the possibility of containing the neighbors of the inserted point), this radius is the radius of the sphere including all neighbors of the first nearest neighbor of the query item. We consider this radius as the one formed by the sum of the distances between the inserted point and its nearest neighbor, and the one between the nearest neighbor and its further neighbor. That is, let consider  $q$  to be the query point and  $v_1$  its nearest neighbor with a distance  $d_1$ . Let consider  $v_2$  to be the furthest neighbor of  $v_1$  with a distance  $d_2$ . The radius  $r$  of the hyper sphere  $SR$  can be then expressed by the following formula:

$$r = (d_1 + d_2)(1 + \epsilon)$$

$\epsilon$  is a relaxation parameter and varies from 0 to 1, it can be initialized according to the state of the data (their dispersion for example) or by the domain knowledge. In what concern us, we have fixed experimentally this parameter to 0.1.

The complexity of the insertion is very low and meets perfectly our starting aims (locating a point in an as short as possible time). It is expressed by:

$$O(2n + n'^3)$$

With  $n(=|V|)$  representing the number of items in the database, and  $n'$  representing the number of items in the hyper sphere ( $\ll n$ ).

This complexity includes the two stages described previously, namely, the search of the radius of the hyper sphere and the recovering of the concerned points which are in it which corresponds to the term  $O(2n)$ . The second term

---

<sup>2</sup> A preliminary and basic version of this method has been proposed in [5].

corresponds to the necessary time for effectively updating the neighborhood relations between the points within the hyper sphere which is very weak taking into account the number of candidates point turned over beforehand<sup>3</sup>. This complexity constitutes the maximum complexity and can be optimized by several ways. The most obvious way is to use a fast nearest neighbor algorithm. We call this method  $\text{LocalInsert}(G(\mathbf{V}, \mathbf{E}), v_i, \epsilon)$  which returns a graph by inserting  $v_i$  into  $G(\mathbf{V}, \mathbf{E})$  with parameter  $\epsilon$ .

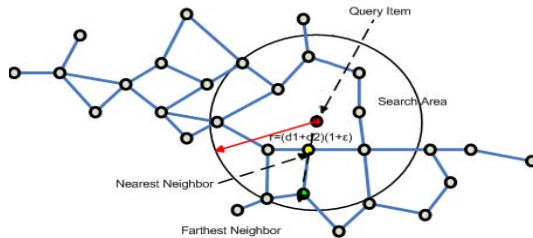
### 3.2 Local Deletion in a Neighborhood Graph

Removing an item from an existing structure is an important task especially when we deal with large databases. Indeed, if we consider an information retrieval system where users interact on line with the indexing structure, the system must be able to not only insert quickly a new item but also remove another within as short as possible time.

In our case, removing an item from an existing graph can be performed in a fast way than the insertion. The principle of this method is rather similar to the insertion method but we can note two main differences. The first one is the fact that this algorithm returns the updated graph with a set of points fewer than the initial one. The second difference (the most important one) is the fact that for the insertion we need to locate the inserted point according to its nearest neighbor, so we need to calculate its first nearest neighbor among the whole dataset which needs  $O(n)$  operations. This operation is omitted from deletion since we know already the necessary information. Thus, the complexity of this task can be expressed by the following formula.

$$O(n + n'^3)$$

With  $n(=|\mathbf{V}|)$  representing the number of items in the database, and  $n'$  representing the number of items in the hyper sphere ( $\ll n$ ). We call this method  $\text{LocalDelete}(G(\mathbf{V}, \mathbf{E}), v_i, \epsilon)$  which returns a graph by deleting  $v_i$  from  $G(\mathbf{V}, \mathbf{E})$  with parameter  $\epsilon$ . Figure 2 illustrates the principle of the proposed methods.



**Fig. 2.** Illustration of the first step of the local updating method(Recovering of the search Area)

<sup>3</sup> The number of candidate points does not exceed a hundred in the worst cases in the performed experiments.

### 3.3 Incremental Neighborhood Graph Construction

As stated in the previous sections, for the incremental construction of neighborhood graphs we propose two different strategies: Forward and Backward. We detail hereafter these two strategies.

1) *Forward strategy.* The objective here is to build the graph (Relative neighborhood graph or Gabriel graph) starting from the set of points and an empty set of edges. That is, the question is how to proceed to build the complete<sup>4</sup> graph quickly using only these information?

To achieve that, the principle of the method is an extension of the local updating method. Indeed, since the proposed method is able to update an existing structure without information loss, it is then easy to generalize the method for accelerating the construction task. So, we start from a graph initialized with two points and one edge, we add one point at each iteration to the graph and we apply the locally updating method on only the set of points composing the graph<sup>5</sup> for locating the neighborhood propagation and updating it.

Algorithm 1 illustrates the different steps of the forward incremental construction strategy.

---

**Algorithm 1.** Forward Strategy

---

**Require:**  $V$ ;  
**Require:**  $\epsilon$ ;  
1: **if**  $|V| < 2$  **then**  
2:   **return**  $G(V, \phi)$ ;  
3: **else**  
4:    $v_1 = \text{Get an item from } V$ ;  
5:    $v_2 = \text{Get an item from } V$ ;  $// v_1 \neq v_2$   
6:    $V = V \setminus \{v_1, v_2\}$   
7:    $V' = \{v_1, v_2\}$   
8:    $E = \{(v_1, v_2)\}$ ;  
9:   **for**  $i = 3$  **to**  $n$  **do**  
10:      $v_i = \text{Get an item from } V$ ;  
11:      $V = V \setminus \{v_i\}$   
12:      $G(V', E) = \text{LocalInsert}(G(V', E), v_i, \epsilon)$ ;  
13:   **end for**  
14: **end if**  
15: **return**  $G(V', E)$ ;

---

The algorithm requires only two elements: A set of points  $V$  containing all the items of the database for which a neighborhood graph will be built, and a

<sup>4</sup> With complete graph we mean a graph which has exactly the same neighborhood than the one we can build with the standard algorithm. In other words, an exact graph not an approximation of the graph.

<sup>5</sup> At this stage, we have two points subsets, the subset containing the points of the graph and the subset containing the remaining points in  $V$ . When we add a point to the graph,  $V$  decreases and the set of points in the graph increases.

relaxation parameter which is used to set up the enlargement importance of the search area according to which the neighborhood updating is propagated.

Note that we suppose here that the neighborhood property (relative neighborhood in this case) is already chosen. So, we start by checking the items count within the provided dataset (Line 1). If the items count is less than two, the algorithm stops. Otherwise we initialize the set of edges with an edge connecting two points randomly taken from  $\mathbf{V}$  (Lines 4 to 7). After that, we apply a local insertion for all the remaining items in the original dataset  $\mathbf{V}$  until it becomes empty. This causes an increasing in the size of the graph as well as in the edges set (Lines 8 to 12). At the end, the algorithm return a graph with the original set of point and a complete neighborhood according to the desired neighborhood property. Note that at the beginning, the algorithm has approximately the same behavior like the standard one. Indeed, in the first iterations, the radius of the hyper sphere is large and thus contains all the used items but as the number of used items is large, the process becomes more interesting and more quickly.

In term of complexity, it is very low and very interesting. it's expressed by the following formula:

$$O(\sum_{i=3}^n (2n_i + n_i'^3))$$

With  $n_i$  representing the number of concerned items at the iteration  $i$ , and  $n_i'$  representing the number of items in the hyper sphere ( $\ll n_i$ ).

2) *Backward strategy.* The objective here is to propose a method able to build (or to deduce) a neighborhood graph (relative neighborhood or Gabriel graph) from an existing structure. To achieve that, let us introduce some preliminaries about two geometrical structures as well as the used properties for deducing the concerned neighborhood graphs (Relative neighborhood graph and Gabriel graph) from a more general structure.

- *Delaunay Triangulation (DT):* Given a set of point  $\mathbf{V}$ , the DT is a particular triangulation, built on  $\mathbf{V}$  which satisfies the empty circum-circle property: The circum-circle (sphere in  $R^3$  or Hyper-sphere in  $R^d$ ) of each simplicial cell in the triangulation does not contain any input point  $v \in V$  [1].
- *Beta-Skeleton:* This type of graph was proposed by Kirkpatrick and Radke [8][6]. It defines a general shape of neighborhood graphs. The neighborhood graph  $U_{v_1, v_2}(\beta)$  for each value of  $\beta$  given ( $1 \leq \beta \leq \infty$ ) is defined like the intersection of two spheres in the following form:

$$U_{v_1, v_2} = B((1 - \frac{\beta}{2})v_1 + \frac{\beta}{2}v_2, \frac{\beta}{2}\delta(v_1, v_2)) \cap ((1 - \frac{\beta}{2})v_2 + \frac{\beta}{2}v_1, \frac{\beta}{2}\delta(v_1, v_2))$$

So, the  $\beta$ -graph of the set of point  $V$  is:

$$U_{v_1, v_2} \cap \mathbf{V} = \phi$$

Note that this structure has the following inclusion property:

$$G_{\beta_1} \subset G_{\beta_2} \subset G_{\beta_3} \subset \dots \subset G_{\beta_n} \text{ with } \beta_1 > \beta_2 > \beta_3 > \dots > \beta_n$$

The relative neighborhood graph and the Gabriel graph are a particular structures of the  $\beta$ -Skeleton. Indeed, a Relative neighborhood graph is a  $\beta$ -graph with  $\beta = 2$  and a Gabriel graph is a  $\beta$ -graph with  $\beta = 1$ . In what concern us, the inclusion properties between the concerned structures are: the neighborhood in the relative neighborhood graph is included in the neighborhood of the Gabriel graph and the neighborhood of the Gabriel graph is included in the neighborhood of the Delaunay Triangulation. This is summarized by the following formula:

$$RNG(\mathbf{V}) \subset GG(\mathbf{V}) \subset DT(\mathbf{V})$$

So, for our backward construction strategy we consider the most general neighborhood structure<sup>6</sup> which is the Delaunay triangulation. The main reason is that the computational complexity of this structure is  $O(n \log n)$  [15].

That is, the backward strategy uses a deletion function to pass from a more general structure to a particular substructure. Algorithm 2 summarizes the different steps of this approach.

---

**Algorithm 2.** Backward Strategy

---

**Require:**  $DT(\mathbf{V}, \mathbf{E})$ ;

**Require:**  $\epsilon$ ;

```

1: for  $i = 1$  to  $n$  do
2:    $v_i = \text{Get an item from } \mathbf{V}$ ;
3:    $DT(\mathbf{V}, \mathbf{E}) = \text{LocalDelete}(DT(\mathbf{V}, \mathbf{E}), v_i, \epsilon)$ ;
4: end for
5:  $G(\mathbf{V}, \mathbf{E}) = DT(\mathbf{V}, \mathbf{E})$ ;
6: return  $G(\mathbf{V}, \mathbf{E})$ ;

```

---

The algorithm requires two parameters: A Delaunay Triangulation, composed by a set of point  $\mathbf{V}$  and a set of edges  $\mathbf{E}$ , and a relaxation parameter  $\epsilon$ . The algorithm processes all the points composing  $\mathbf{V}$  by applying a locally deletion function. So, for each considered point, we calculate the possible propagation of the neighborhood and we apply a neighborhood property in the recovered area. Note here that when we apply the `LocalDelete` function we don't remove the concerned items from the dataset but we remove only the edges which do not satisfy the considered neighborhood property.<sup>7</sup>

The complexity of the algorithm can be expressed by the following formula:

$$O(\sum_{i=3}^n (n_i + n'^3))$$

With  $n_i$  representing the number of concerned items at the iteration  $i$ , and  $n'$  representing the number of items in the hyper sphere ( $\ll n_i$ ).

---

<sup>6</sup> We are concerned into three used structures in this paper (RNG, GG, DT).

<sup>7</sup> A function for removing an edge has the same behavior that the one for removing a point; it is easy to modify `LocalDelete` and replace the deletion of a point by a deletion of an edge.

## 4 Experiments and Results

### 4.1 Validity of the Obtained Results

The validity and the utility of the local updating method in its basic version has been shown in [5] and in [4] in its current version. The objective here is to show the validity and the utility of the obtained results for the generalization of the method (the incremental construction of the graph) as well as for the deletion task. So, the final objective is to show that the graph built using our algorithm is exactly the same as the one we can build using the classical algorithm.

To achieve that, we have made experiments to check the structural differences between a classical graph and a graph built with our algorithm. The two graphs are built and compared on the same dataset at each iteration. We generated for that different datasets with different items count and different dimensions. Table 1 summarizes the obtained results.

**Table 1.** Correspondence between two graphs

Dataset	Items count	Dimension	% corresp. $G_1$ and $G_2$
1	1.000	15	100%
2	1.000	25	100%
3	2.500	25	100%
4	2.500	30	100%
5	5.000	45	100%
6	5.000	60	100%

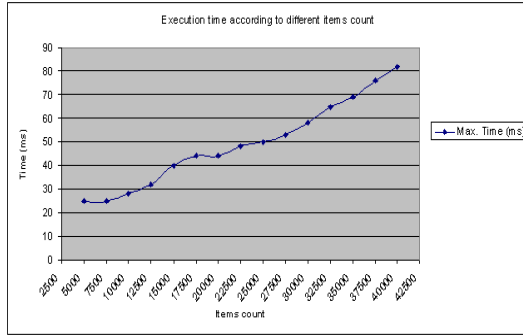
To calculate the correspondence degree between two graphs, each node in the graph is identified by a unique identifier. So, for each node in the first graph we check if all its neighbors are also the neighbors of the same node in the second graph and if the two nodes have the same nodes count. The obtained results are satisfactory and show a complete correspondence between the two graphs.

### 4.2 Scalability of the Method

The execution times of the locally updating method were evaluated in [5]. Here, we are interested in the evaluation of the scalability of the method since modern databases have the property to be large. So, we consider here the size of the database and the objective is to show that the response times are not very variable when the size of the database grows.

The evaluation protocol is as follows: we generate an artificial dataset with a dimension  $d = 40$ . We start with an initial dataset with 5.000 items. We increase the number of items with 2.500 items until reaching 40.000 items. In each iteration, 10 items are taken arbitrary, inserted in the graph and the response times are recovered. The graphic of Figure 3 illustrates the obtained results in each iteration by representing the maximum of the execution time of each experience.





**Fig. 3.** Illustration of the scalability of the proposed method

Figure 3 shows the times evolution (maximum) according to a variable number of items. Times are expressed in *milliseconds*. The results are very interesting. Indeed, increasing the volume of the database does not affect considerably the execution times. We can particularly note that the execution times always remains in values expressed in milliseconds. In addition let us point that we did not implement any optimization of the  $k - NN$  algorithm. Indeed, more optimized implementation of this algorithm can improve the results considerably. So, with these results, we can appreciate the proposed method, not only for the quality of the results and the response times [5], but also for its scalability.

The results shown beforehand illustrate the interest of the locally updating method<sup>8</sup>. In order to show the interest of the incremental construction approach and to illustrate the benefits compared to the standard algorithm we performed further experiments which we describe hereafter.

We generate artificial datasets with a dimension  $d = 50$  and different items count 5.000, 10.000, 20.000, 40.000, 50.000, 75.000. We apply the standard algorithm on the datasets to build a relative neighborhood graph and we recover the execution times. After that we use our incremental algorithm<sup>9</sup> to build the same graph on the same datasets. The obtained results<sup>10</sup> are illustrated on Figure 4.

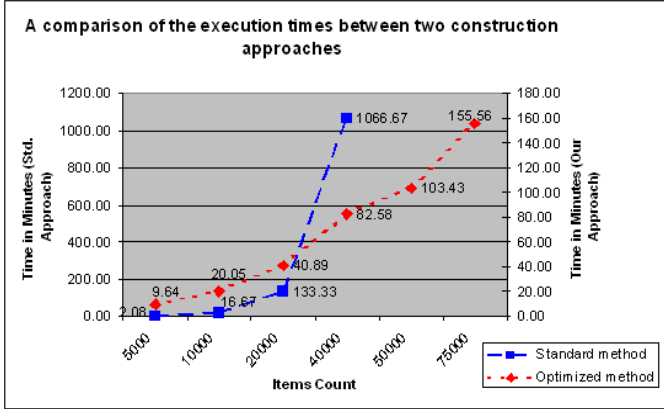
The graphic shows the evolution of the construction time of two methods: The standard method (dashed line corresponding to the left axis) and our method (dotted line corresponding to right axis). At the beginning, the two methods have almost the same execution time (Until 10.000 items), but after that, the gain obtained using our algorithm becomes very important.

These results confirm the interest of the proposed approach. Indeed, in addition to the scalability of the updating tasks (insert and delete an item in the graph),

<sup>8</sup> These results can illustrate also the deleting method since it is also based on the same principle like the insertion method.

<sup>9</sup> We tested here the Forward approach.

<sup>10</sup> The represented times on the graphic are a theoretical results, the real results have exactly the same behavior like these ones. We prefer to represent the theoretical results because the implementation of the incremental algorithm is not yet optimized.



**Fig. 4.** A comparison of execution times between the standard method and the optimized method

the proposed approach is able to build a graph using large databases within reasonable time. This point is very important because of the largeness of the today databases.

## 5 Conclusion and Future Work

The direct use of neighborhood graphs is not suitable, because their complexity does not make it reasonable to build them starting from great databases. We proposed in this article an effective graph based index structure and this by introducing and optimizing all the necessary operations of an index (build the index, insert an item into the index, delete an item from the index, and quickly retrieve the index). So, we discussed two optimization strategies: Forward and Backward strategies. These strategies are efficient for improving the performances of the usage of these structures especially in a retrieval context and in a situation where the databases are large.

As a future work, we plan to fix the problem of the relaxation parameter determination by setting up an automatic determination function. This can be done by taking into account some statistical indicators on the data like the dispersion. Also, we plan to optimize more the computational complexity of the basic methods (insertion and deletion) for reducing the database scans.

## Acknowledgments

The authors are grateful to Prof. Zighed and Prof. Tanaka for their support to conduct this work. This work was partially supported by Région Rhône Alpes under grant EMERGENCE 2004, France, and Core-to-Core Program (No.18001) by JSPS in Japan, and the grant-in-aid for scientific research (No.18700131) funded by MEXT, Japan.

## References

1. P. Cignoni, C. Montani, and R. Scopigno. Dewart: A fast divide and conquer delaunay triangulation algorithm in ed. *Computer-Aided Design*, 30(5):333–341, 1998.
2. K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic zoology*, 18:259–278, 1969.
3. V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170–231, 1998.
4. H. Hacid and D. Zighed. Content-based image retrieval in large image databases. In *IEEE International Conference on Granular Computing (GrC 2006)*, Atlanta, USA, May 2006.
5. H. Hacid and D. A. Zighed. An effective method for locally neighborhood graphs updating. In *DEXA*, pages 930–939, 2005.
6. J. Jaromczyk and G. Toussaint. Relative neighborhood graphs and their relatives. *P-IEEE*, 80:1502–1517.
7. J. Katajainen. The region approach for computing relative neighborhood graphs in the lp metric. *Computing*, 40:147–161, 1988.
8. D. G. Kirkpatrick and J. D. Radke. A framework for computational morphology. In *Computational Geometry*, G. T. Toussaint, ed., Elsevier Science Publishers, North Holland, pages 217–249, 1985.
9. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1, pages 281–297, 1967.
10. F. Preparata and M. I. Shamos. *Computational Geometry-Introduction*. Springer-Verlag, New-York, 1985.
11. W. D. Smith. Studies in computational geometry motivated by mesh generation. *PhD thesis, Princeton University*, 1989.
12. P. Somervuo and T. Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10(2):151–159, 1999.
13. G. T. Toussaint. The relative neighborhood graphs in a finite planar set. *Pattern recognition*, 12:261–268, 1980.
14. G. T. Toussaint. Some insolved problems on proximity graphs. D. W. Dearholt and F. Harrary, editors, *proceeding of the first workshop on proximity graphs. Memoranda in computer and cognitive science MCCS-91-224. Computing research laboratory. New Mexico state university Las Cruces*, 1991.
15. E. Welzl, P. Su, and R. L. S. D. III. A comparison of sequential delaunay triangulation algorithms. *Comput. Geom.*, 7:361–385, 1997.

# Learning Network Topology from Simple Sensor Data

Dimitri Marinakis, Philippe Giguère, and Gregory Dudek

Centre for Intelligent Machines, McGill University,  
3480 University St, Montreal, Quebec, Canada H3A 2A7  
{dmarinak, philg, dudek}@cim.mcgill.ca  
<http://www.cim.mcgill.ca>

**Abstract.** In this paper, we present an approach for recovering a topological map of the environment using only detection events from a deployed sensor network. Unlike other solutions to this problem, our technique operates on *timestamp free* observational data; *i.e.* no timing information is exploited by our algorithm except the ordering. We first give a theoretical analysis of this version of the problem, and then we show that by considering a sliding window over the observations, the problem can be re-formulated as a version of set-covering. We present two heuristics based on this set-covering formulation and evaluate them with numerical simulations. The experiments demonstrate that promising results can be obtained using a greedy algorithm.

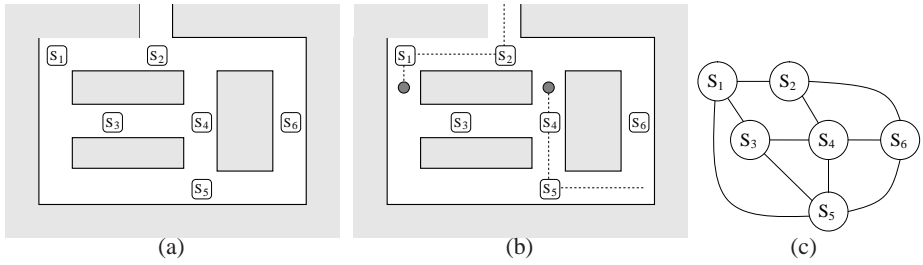
**Keywords:** sensor networks, topology inference, set-covering, mapping.

## 1 Introduction

In this paper we consider the problem of learning the *topology* of a network of sensors through the exploitation of motion in the environment. We assume that an individual sensor is capable of detecting the passage of an agent through its local region, but is unable to generate a reliable signature. Our approach uses the combined observational data returned from our network to infer the topological relationships between the sensors.

We will illustrate the problem with a simplified example. Figure 1(a) depicts a sensor network distributed within an indoor environment. Let us assume that the network has been deployed for some purpose, such as surveillance, and requires knowledge of the inter-node connectivity in order to fulfill its function. During some initial calibration period the network collects observations of agents passing by each sensor (Figure 1(b)). The problem we are trying to solve is how to use these collected observations to construct the topological description of the network shown in Figure 1(c). This type of network might arise if wireless cameras were deployed in a workplace environment.

In this work, we assume not only that the agents moving through the environment are indistinguishable, but that there are no temporal clues that can be used to aid the inference process. In other words, the detection events are correctly ordered but are not *time-stamped*. Therefore, when our inference algorithm is employed, the time-stamp data can be discarded or simply not collected in the first place. In order to exploit timing information in the observational sequence some model of agent motion in the environment needs to be either constructed based on prior assumptions, or learned from the



**Fig. 1.** An example of a sensor network which we wish to calibrate. a) The original ad-hoc deployment. b) An Example of agent motion exploited by the calibration process. c) The desired topological connectivity map of the network.

data. Our technique, however, allows the correct edges in the graph to be inferred while avoiding the prior domain knowledge or algorithmic complications involved in constructing an adequately accurate motion model. By employing a sliding window over the observations, we will show that the problem can be re-formulated as a version of the well understood set-covering problem and accurate results can be obtained without timing information.

The ability of a surveillance or monitoring system to automatically determine the connectivity parameters describing its environment is useful for a number of reasons. Although the topology information can be manually entered during installation, more detailed parameters such as the relative connectivity strength between links are difficult to determine, and a change in the environment or network would require re-calibration. Once calibrated, the connectivity information could aid in conventional target tracking and additional monitoring activities. For example, by reconstructing trajectories, a vehicle monitoring network distributed about a city could help make decisions about road improvements which might best alleviate congestion. In addition, the topological information could be combined with relative localization techniques [1] [2] [3] [4] to recover a more complete representation of the environment.

## 2 Background

Although the topological mapping problem has been well explored in mobile robotics [5] [6] [7] [8], most sensor network related investigations have been more recent [9] [10] [11] [12]. The outcome is generally a graph where vertices represent embedded sensors in the region and edges indicate navigability.

Ellis, Markis, and Black [9] approached the topology inference problem in the context of camera-based sensing. Their technique exploits temporal correlations in observations of agents' movements. They outlined an approach in which they first identified entrance and exit points in camera fields of view to generate a graph from video data. They then used a thresholding technique to look for peaks in the temporal distribution of travel times between entrance-exit pairs; a clear peak suggesting that the cameras are linked. This approach requires a large number of observations, but does not rely on

object correlation across specific cameras. Thus, the method can be used to efficiently produce an approximate network connectivity graph.

Marinakakis and Dudek [11] [12] have recently presented a solution to the topological inference problem that is based on a stochastic version of the Expectation Maximization algorithm. Their approach uses only detection events from the deployed sensors and is based on reconstructing plausible agents trajectories. Results presented from simulations and experimental data suggest that their technique produces accurate results under a variety of conditions and compares well to other approaches.

When the observation are information-poor, topology inference through trajectory reconstruction has much in common with the data association problem in multi-object tracking and similar statistical techniques are employed. For example, in [13], event detections alone were used for the tracking of multiple targets using Markov Chain Monte Carlo (MCMC). Similarly, [14] approached a traffic monitoring problem using limited sensor data observations through a stochastic sampling approach.

A key observation regarding all of the approaches mentioned above is that their performance will suffer if temporal information is removed from the observations. Ellis, Markis, and Black rely explicitly on this temporal data, while the approaches employing probabilistic frameworks [11] [13] [14] exploit the delay information to aid in the data association problem.

In the remainder of this paper, we consider the problem of solving the topology inference problem, relying only on the ordering of the timing information in the observational data. We discuss theoretical aspects of this version of the problem and present an algorithm for its solution. It is our hope that concepts presented here can be incorporated into more general techniques for topology inference, or used in their own right.

### 3 Problem Definition

We formulate the problem of learning the network topology as the inference of a directed graph  $G = (V, E)$ , where the vertices  $V = v_i$  represent the locations where sensors are deployed, and the edges  $E = e_{i,j}$  represent the connectivity between them; an edge  $e_{i,j}$  denotes a path from the position of sensor  $v_i$  to the position of sensor  $v_j$ . The sources of motion in the sensor network are modeled as some number  $N$  of agents moving asynchronously through the graph. Each agent generates an observation every time it visits a vertex. This corresponds to an agent passing near a particular sensor which then detects the presence of motion in its region.

The input to the problem is an ordered list of observations  $O = \{o_t\}$ , each of which is identifiably generated by one of the sensors; *i.e.* each  $o_t \in [1, V]$ . The goal is to find the correct underlying graph  $G$  explaining this observational sequence.

## 4 Algorithm Formulation

### 4.1 Smallest Graph Is Correct Answer

The key idea behind our approach is to find the *smallest*<sup>1</sup> graph that successfully explains the observed data. Leaving aside for the moment the actual implementation

---

<sup>1</sup> The graph with the smallest number of edges.

details, let us consider this idea in more depth by proposing the existence of an algorithm  $A$  that takes as an input the assumed number of agents  $N'$  in the environment and the observational sequence and returns as an output the smallest graph consistent with the observations.

Our algorithm  $A$  considers each of the possible trajectories that could be taken by these  $N'$  agents given the observational sequence and then selects the trajectory set that requires the smallest number of inter-vertex traversals. The algorithm then returns the graph populated only with edges that correspond to the inter-vertex traversals required by this chosen trajectory set.

The concept that the simplest solution explaining the data is probably the correct solution has been used successfully in a different version of the topology inference problem [12]. The principle, known as Occam's razor, states, "if presented with a choice between indifferent alternatives, then one ought to select the simplest one." The concept is a common theme in computer science and underlies a number of approaches in AI; *e.g.* hypothesis selection in decision trees and Bayesian classifiers [15]. We will show in the next section that under certain assumptions, we can prove that an algorithm trying to find the smallest graph will return the correct answer.

## 4.2 Correctness of the Smallest Graph Assumption

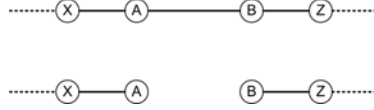
In this section we present a proof that the smallest graph  $G$  consistent with the observations is the correct solution  $G_c$  given the following assumptions:

1. There are an infinite number of observations,  $O$ .
2. The motion of each of the agents is random.
3. The true number of agents in the system  $N$  is fixed and bounded by the number assumed by the algorithm  $A$ ; *i.e.*  $N \leq N'$ .
4. The transit time between nodes is un-bounded.
5. There are no self-referential connections in the true graph  $G_c$ ; *i.e.* no agent may trigger two observations by one passage through the region of a single sensor.

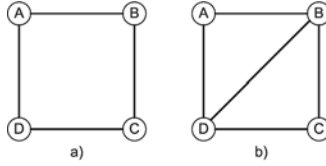
## 4.3 Proof of Smallest Graph

First, we will prove that there exist no graph smaller than  $G_c$  that can explain the observed data, given the above stated assumptions. This is done by showing that it is possible to have sequences generated by  $G_c$  that cannot be explained by this smaller graph  $G'_c$ . In other words,  $G'_c$  is not consistent with the observations, and by definition cannot be a solution.

Let us consider a graph  $G'_c$  created by removing a single edge from  $G_c$ , as in figure 2. In this case, we remove the edge  $AB$  from graph  $G_c$ . Let us now create a valid observational sub-sequence  $O = ABABABAB...AB$  which was created in truth by a single agent traversing back and forth on the edge  $AB$ . The only way agents in a graph  $G'_c$  could generate this observational sequence would be if some number of them were 'stationed' at node  $X$ , and some number 'stationed' at node  $Z$ , and alternatively one agent from  $X$  traversed the edge to  $A$ , and then one from  $Z$  traversed the edge to  $B$ . However, if the length,  $|O|$  of the observational sub-sequence  $O$  is larger than the



**Fig. 2.** Example of removing edge  $AB$  from graph  $G_c$ , (shown partially on top), to create graph  $G'_c$ , (shown partially below)



**Fig. 3.** a) The correct graph  $G_c$  b) an incorrect graph

maximum possible number of agents  $N'$ , then there will not be enough agents in  $G'_c$  to generate  $O$ . Therefore, the edge  $AB$  must be present in any consistent solution. Applying this to all the edges in  $G_c$ , we see that a solution that can explain all the transitions must have at least all the edges in  $G_c$ . Consequently, the smallest consistent graph is the correct graph  $G_c$ .

Note that this analysis requires that there be both an infinite number of observations and random motion on the part of the agents in order to allow such very rare observational sequences to exist. However, this concept holds with less formality to very large sequences of observations. It becomes less likely for a graph to successfully explain an observation sequence while missing portions of the real graph as the number of observations increases.

#### 4.4 Impact of Estimated Number $N \leq N'$ of Agent on Solution $G$

In the following sections, we will show the impact of the number  $N$  of agent used to find a solution. More precisely, we will show that if an algorithm overestimates the number  $N$  of agents in the system, it will still give the correct answer  $G_c$ , but not if it underestimates it.

**Lemma 1.** *Overestimating the number  $N$  of agents while looking for the smallest graph results in the correct solution.*

To prove this lemma, we will show that a path generated by a single agent can be *spliced* between two agents using a ‘tag team’ method, and yet will still yield a the correct graph  $G_c$ . That way, all superfluous agents used in the algorithm can be ‘hidden’ by splicing a valid path repeatedly.

Let us consider a true sequence of vertex traversals  $S$  generated by a single agent. First, without loss of generality, select any vertex  $v_{splice} \in S$  as a splicing node. We can now pair any two *virtual* agents together to generate this traversal sequence  $S$  in the following way. Let one of the virtual agent be initially stationed at  $v_{splice}$ . When



the other virtual agent enters this vertex it will exchange its role with the first agent, as in a game of tag team wrestling. The other agent will now leave the vertex  $v_{splice}$ , generating a sub-sequence of  $S$  until it re-enters  $v_{splice}$ , where again they will switch roles.

As an example, let us consider the vertex sequence  $S = ABCDADC D ABCBA$  generated by a true agent in  $G_c$  of Figure 3(a.). We choose  $v_{splice} = C$ . Now the vertex sequence  $S$  assumed to come from a single agent looks like the following:  $ABcdadCDABcba$  where capital letters are used for the path  $P_1$  of agent one, and small bold letters are used for the path  $P_2$  of agent two. The individual virtual sequences  $P_1 = ABCDAB$  and  $P_2 = cdadcba$  are both valid sequences in the correct graph  $G_c$ .

This analysis shows that we can assume the existence of more agents than the number that actually generated the observation sequence, and still produce paths that are consistent with the correct graph.

**Lemma 2.** *Underestimating the number of agents can result in an incorrect solution.*

To prove this lemma, we will simply show that there exists at least one observational sequence  $O$  such that underestimating the number of agents creates a false solution. Let us consider again the graph depicted in figure 3(a.) and let us consider the motion of two agents in this graph. Agent one will follow the path  $ABCDADC$ , and agent two will follow the path **db**. By combining the two paths, it is possible to get the sequence of observations:  $O_1 = ABdCDAbDC$ . If we assumed the existence of only one agent, then the smallest graph that can explain the transitions  $O_1$  is displayed in figure 3(b.) and is incorrect.

Using the above stated two lemma, we arrive at the following theorem (which holds true given the earlier stated assumptions):

**Theorem 1.** *If the number of assumed agents  $N'$  in the algorithm is equal or greater than the true number of agents  $N$  that generated the observations, then the smallest graph consistent with the observations will be the correct graph  $G_c$ . If the number of assumed agent is smaller than the true number, then there are no guarantees that the smallest graph consistent with the observations will be the correct graph  $G_c$ .*

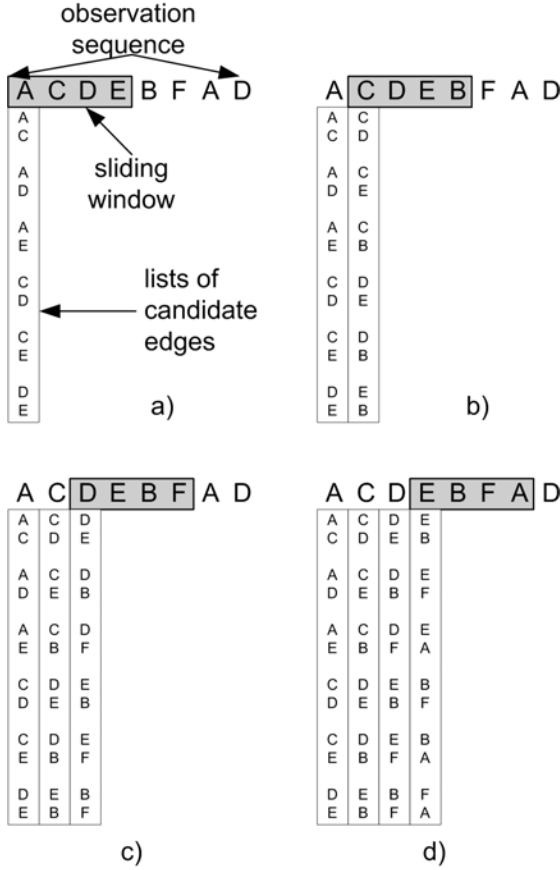
In the next section, we draw on this theoretical analysis to motivate a pragmatic approach for topology inference.

## 5 The Sliding Window Approach

We now present an algorithm for estimating the smallest possible graph given an observation sequence. Our approach is based on the following lemma:

**Lemma 3.** *In any given continuous sequence of  $N_S > N$  observations, at least  $(N_S - N)$  transitions between observations correspond to edges in the correct graph  $G_c$ .*

For example, let  $N_S = 4$  and  $N = 3$  and the recorded observational sequence be  $ABCD$ . The possible transitions between nodes are  $AB$ ,  $AC$ ,  $AD$ ,  $BC$ ,  $BD$  and  $CD$ .



**Fig. 4.** Example of generating candidate edges for each sliding window position. The window is moved to the right from a) to d).

Since we have one more observation than the number of agents, ( $N_S - N = 1$ ), it means that at least one agent must have generated more than one of the observations in this sequence, and therefore at least one of the transitions listed above must be valid; *i.e.* present in  $G_c$ . Note that the number of potential transitions generated with a sequence of  $N_S$  observations is:

$$N_T = \frac{(N_S - 1)N_S}{2}$$

Our technique is to employ a sliding window of size  $N_S = N + 1$  to consider in turn small continuous subsequences of the entire observation sequence  $O$ . Each of these subsequences gives rise to a list of  $N_T = (N^2 + N)/2$  candidate edges  $L_i$ , one of which must be present in the true solution  $G_c$ . Once the window has moved over the complete observation sequence  $O$ , there will be  $K = |O| - N_S$  lists generated. Figure 4 shows an example of generating candidate edges using the sliding window approach. Motivated

by Theorem 1, our approach is find the smallest graph that can explain at least one edge in each in each of these candidate lists:  $L_1, L_2, \dots, L_K$ .

This problem can be shown to be equivalent to the set-covering problem which is NP-complete, however, several heuristics can be employed to estimate an optimal solution. We will consider a two heuristic approaches in the next sections.

## 5.1 A Greedy Approach

One method of obtaining a solution to the sliding window problem posed above, is to adopt a greedy approach. This is a standard heuristic often used with good results for set-covering problems. In our domain, the greedy algorithm would work as follows:

1. Begin by marking all candidate lists  $L_1, L_2, \dots, L_K$  unexplained and initialize a list of edges  $E$  to be empty.
2. Find the edge  $e$  that is present in the greatest number of currently unexplained candidate lists.
3. Remove from consideration those candidate lists which contain edge  $e$  by marking them explained, and add  $e$  to  $E$ .
4. Repeat steps 2 to 3 until all lists are marked explained. Return the graph corresponding to our list of edges  $E$  as the final solution.

## 5.2 A Statistical Approach

A statistical approach could also be used to determine the correct edges in  $G_c$ . The number of times a given edge has been seen in any candidate list could be tallied up. Those edges that occur with a frequency greater than some threshold  $T$  could then be selected.

Let us consider a suitable value for the threshold  $T$ . If  $G_c$  corresponded to a fully connected undirected graph, the average tally of each edge would be:

$$\mu = \frac{KN_T}{|E|}$$

where  $N_T$  is the number of candidate edges generated *per* window,  $K$  is the number of candidate lists (windows), and  $|E|$  is the number of potential edges in the graph. Replacing  $K$  with  $|O| - N_S$ ,  $N_T$  with  $(N^2 + N)/2$ , and  $|E|$  with  $V(V - 1)$ , we arrive at:

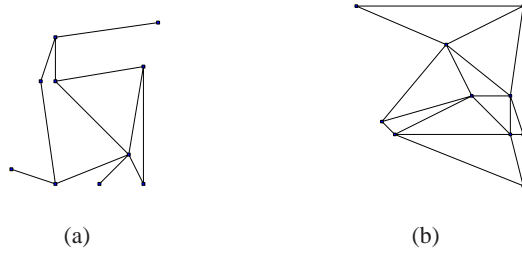
$$\mu = \frac{(|O| - N_S)(N_S - 1)N_S}{V(V - 1)}$$

Since we expect  $G_c$  to contain less edges than its fully connected counterpart,  $T = \mu$  can be expected to be a suitable threshold.

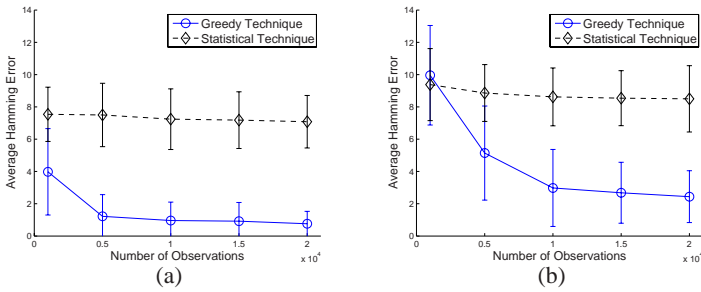
# 6 Experiments

## 6.1 Simulator

We have examined the sliding window approach with a number of experiments conducted in simulation. We have constructed a simulation tool that takes as input a graph and the



**Fig. 5.** Example of graphs created using the Delaunay triangulation technique: a) 10 node graph with 12 edges, b) 10 node graph with 20 edges

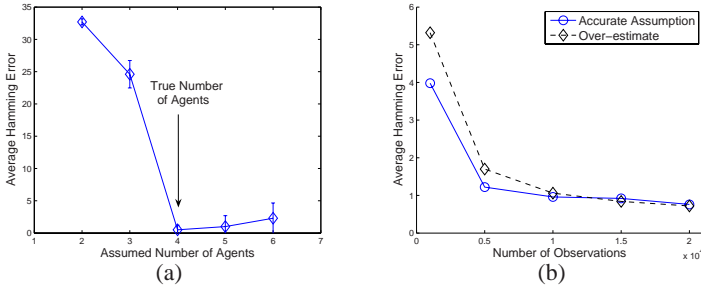


**Fig. 6.** Mean Hamming Error obtained from the two techniques for various numbers of observations averaged over 50 randomly produced graphs. (Error bars show one standard deviation). Results obtained from 10 node graphs with: a) 12 edges b) 20 edges.

number of agents in the environment and outputs a list of observations generated by randomly walking the agents through the environment. A number of experiments were run using this simulator on randomly generated planar, connected graphs (Figure 5). The graphs were produced by selecting a connected sub-graph of the Delaunay triangulation of a set of randomly distributed points. For each experiment, the results were obtained by considering the Hamming error between the true and inferred graph.

## 6.2 Results

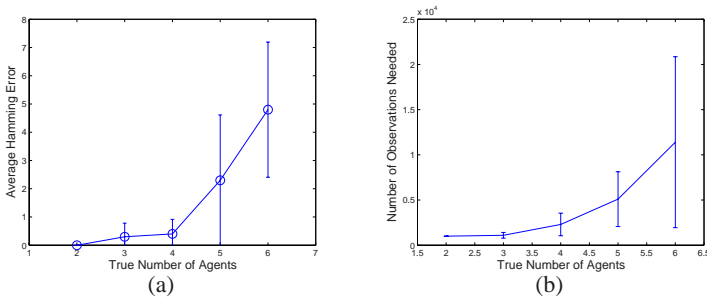
The greedy approach was capable of producing accurate results in moderately sized graphs with a reasonable number of agents, given an adequate number of observations. Although not as accurate on average as the greedy approach, the statistical approach was also capable of producing a solution near the true answer. Figure 6 compares the accuracy of these two approaches over 50 randomly produced graphs of 10 nodes and two different edge densities. Note that the accuracy of both approaches tended to increase as the number of observations increased. It also appeared that denser graphs required larger numbers of observations to obtain the same accuracy level than that obtained in sparser graphs. Additionally, it was observed that the greedy approach obtained a better Hamming error on average for less dense graphs. However, when the *proportion* of the



**Fig. 7.** Results obtained by differing the assumed number of agents for graphs of size 10 nodes and 14 edges. a.) Hamming Error as a function of the assumed number of agents for the greedy algorithm. Results obtained with 10000 observations generated from 4 agents and averaged over 10 graphs. (Error bars show one standard deviation). b.) Mean Hamming Error as a function of observations for an accurate assumption of 4 agents and an over-estimate of 5 agents. Results averaged over 50 graphs.

true graph structure recovered was considered, this effect was lessened. For example, for the experiment shown in figure 6, the Hamming error divided by the true number of edges in the graph was approximately double for denser graph, while the Hamming error alone was approximately triple.

Unsurprisingly, the accuracy of the statistical approach was very sensitive to the value of the threshold selected. Experiments not shown here verified that the value for  $T$  selected above was generally suitable for graphs of various densities and sizes, although often better results could be obtained for any specific graph type through careful tuning. This approach requires relatively little computational effort and might have value as a bootstrapping technique for more complex approaches such as the one presented in [11].



**Fig. 8.** Performance of algorithm as a function of the true number of agents for the greedy algorithm where the assumed number of agents is set to the correct number. Results averaged over 10 graphs of size 10 nodes and 14 edges; (error bars show one standard deviation). a.) Hamming Error obtained with 10000 observations. b.) Number of observations required to obtain a result with a Hamming error of 2 or less.

As predicted by theorem 1, the effect of over-estimating the number of agents in the environment was indeed less detrimental than that of under-estimating the number of agents. Figure 7 shows the result of assuming various numbers of agents in one situation for the greedy approach. As the over-estimation increases, the required number of observations needed to solve the problem also increases.

The problem of topology inference becomes more difficult as more agents are added to the system. Figure 8 shows the correspondingly poorer performance obtained with the greedy approach on the same set of graphs with observations generated from different numbers of agents. Even if the correct number of agents is known, we suspect that less information is available as the number of agents increases. As the size of the sliding window increases, so does the number of candidate edges generated by each sliding window. Therefore, the ratio of known correct to incorrect edges decreases, and hence, more observations are needed to obtain the same level of error.

## 7 Conclusion

In this paper, we have described a way of learning the *topology* of a sensor network, using only event ordering information. We presented a theoretical analysis of the problem, and re-formulated it as a set-covering problem. Two methods were presented to solve this problem, one based on statistics, and one based on a sliding window technique. We explored the effectiveness of both approaches through numerical simulations for various test cases. Our work demonstrates the promise of this approach for topology inference.

In future work, we hope to extend some of our theoretical results. We would be interested in furthering our understanding regarding the impact of the number of agents in the system since this value tends to dilute the information gained per each observation. This effort would entail deriving a relationship for the information gained in the context of the sliding window approach. Additionally, it would be of interest to find an analytical relationship between the number of observations needed to solve a problem and the corresponding density of agents in the system; *i.e.* the ratio of agents to edges in the true graph. On a different level, we would like to apply the findings presented in this paper to the version of the problem where timing data is available and compare it to other established methods.

**Acknowledgments.** We would like to acknowledge Ketan Dalal for his helpful analysis of this problem.

## References

1. Savvides, A., Han, C., Strivastava, M.: Dynamic fine-grained localization in ad-hoc networks of sensors. In: 7th annual international conference on Mobile computing and networking, Rome, Italy (2001) 166–179
2. Moore, D., Leonard, J., Rus, D., Teller, S.: Robust distributed network localization with noisy range measurements. In: Proc. of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys '04), Baltimore (2004)
3. Marinakis, D., Dudek, G.: Probabilistic self-localization for sensor networks. In: AAAI National Conference on Artificial Intelligence, Boston, Massachusetts (2006)

4. Ihler, A.T., Fisher III, J.W., Moses, R.L., Willsky, A.S.: Nonparametric belief propagation for self-calibration in sensor networks. *IEEE Journal of Selected Areas in Communication* (2005)
5. Shatkey, H., Kaelbling, L.P.: Learning topological maps with weak local odometric information. In: *IJCAI97, San Mateo, CA (1997)* 920–929
6. Choset, H., Nagatani, K.: Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation* **17**(2) (2001) 125 – 137
7. Remolina, E., Kuipers, B.: Towards a general theory of topological maps. *Artif. Intell.* **152**(1) (2004) 47–104
8. Ranganathan, A., Dellaert, F.: Data driven MCMC for appearance-based topological mapping. In: *Proceedings of Robotics: Science and Systems, Cambridge, USA (2005)*
9. Makris, D., Ellis, T., Black, J.: Bridging the gaps between cameras. In: *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2004, Washington DC (2004)*
10. Rekleitis, I., Meger, D., Dudek, G.: Simultaneous planning localization, and mapping in a camera sensor network. *Robotics and Autonomous Systems (RAS) Journal, special issue on Planning and Uncertainty in Robotics* (2006)
11. Marinakis, D., Dudek, G.: A practical algorithm for network topology inference. In: *IEEE Intl. Conf. on Robotics and Automation, Orlando, Florida (2006)*
12. Marinakis, D., Dudek, G.: Topological mapping through distributed, passive sensors. In: *International Joint Conference on Artificial Intelligence, Hyderabad, India (2007)*
13. Songhwai Oh, Phoebus Chen, M.M., Sastry, S.: Instrumenting wireless sensor networks for real-time surveillance. In: *Proc. of the International Conference on Robotics and Automation.* (2006)
14. Pasula, H., Russell, S., Ostland, M., Ritov, Y.: Tracking many objects with many sensors. In: *IJCAI-99, Stockholm (1999)*
15. Michell, T.M.: *Machine Learning*. McGraw-Hill, Boston (1997)

# Reinforcement Learning in Nonstationary Environment Navigation Tasks

Terran Lane, Martin Ridens, and Scott Stevens

Department of Computer Science, University of New Mexico  
`terran,mridens,scottish@cs.unm.edu`

**Abstract.** The field of reinforcement learning (RL) has achieved great strides in learning control knowledge from closed-loop interaction with environments. “Classical” RL, based on atomic state space representations, suffers from an inability to adapt to nonstationarities in the target Markov decision process (i.e., environment). Relational RL is widely seen as being a potential solution to this shortcoming. In this paper, we demonstrate a class of “pseudo-relational” learning methods for nonstationary navigational RL domains – domains in which the location of the goal, or even the structure of the environment, can change over time. Our approach is closely related to deictic representations, which have previously been found to be troublesome for RL. The key insight of this paper is that navigational problems are a highly constrained class of MDP, possessing a strong native *topology* that relaxes some of the partial observability difficulties arising from deixis. Agents can employ local information that is relevant to their near-term action choices to act effectively. We demonstrate that, unlike an atomic representation, our agents can learn to fluidly adapt to changing goal locations and environment structure.

## 1 Introduction

The field of reinforcement learning (RL) has made great progress in learning control knowledge for tasks as diverse as robotic manipulator control, elevator scheduling, game playing, navigation, network packet routing, and autonomous flight. Yet there remain a number of large open questions in both theory and practice of RL. A key problem is dealing with changing tasks – that is, adapting to changes in the system dynamics or reward functions without needing to re-learn a model or policy from scratch. This problem can be posed in terms of environmental nonstationarities – changes in the underlying Markov decision process (MDP) in which the agent is acting. In this paper, we tackle a limited form of this problem by examining adaptation to nonstationarity in the context of autonomous navigation. We claim that the restriction to spatial environments supports adaptation by allowing the agent to exploit the topology of the environment.

The core issue is that “classical” RL algorithms are rooted in an *atomic state space* representation: the agent learns a policy mapping from unique state IDs



onto its action set. This representation ties the policy closely to a specific MDP and prevents adaptation, because there is no way to translate state IDs between different MDPs. Recently, there has been great interest in *relational representations* to address this problem. We employ a very limited form of relational representation in which the agent has a bounded number of relations that describe its immediate surroundings and its relationship to the goal location. This fairly natural “agent-centric” representation allows the agent to adapt to changes in goal location and to handle switching among a fixed set of environments. Such an approach is most closely related to a *deictic* representation [1], in which the agent has a bounded number of “pointers” to objects<sup>1</sup>. Previous authors have had difficulties with deictic representations because they are fundamentally partially observable and lead to state aliasing – a double-edged sword that both supports knowledge generalization and can “confuse” the agent by requiring different actions for the same percept [2].

While Finney *et al.* found that deixis was ill-suited to a blocks-world domain, we find that a deictic-like representation provides substantial benefits in the navigation problems we examine. We show that our approach can achieve two different important kinds of adaptation to nonstationarity: handling goal relocation in a fixed environment and handling changes to the environment. We attribute this success to the nature of the problems that our agents face – navigational tasks in worlds with a *topology* based on a metric that bounds the possible outcomes of actions and the possible trajectories open to the agent. The constraints imposed by the metric allow the agent to navigate “well” (albeit not optimally) with local information.

The goal of this paper is not to introduce a new learning algorithm, but rather to study the effects of representational choices and the topology of the environment on an already well-understood RL algorithm. Our contribution is to demonstrate the strengths and weaknesses of this representation in spatial environments and to set it in contrast to existing work on deixis in blocks-world environments. To isolate the effects of our representation, we omit techniques like function approximation that are necessary for large state-space tasks, thus restricting our agents to relatively small environments. Nonetheless, the nonstationary environment problem we study is equivalent to a partially-observable Markov decision process with one hidden variable and over three hundred thousand states.

## 2 Background

Adapting in the face of nonstationary Markov decision processes (MDPs) is an instance of the more general *RL knowledge generalization* problem. There are two broad research directions attempting to address this problem: function approximation [3,4,5] and relational representations [1,6,7,8]. In the former approach,

---

<sup>1</sup> A deictic representation is an agent-centric knowledge representation in which the agent reasons about the world not in terms of an absolute coordinate frame (*the thing at location  $(x, y)$* ), but in terms of referential pointers (*the thing that I am pointing to* or *the thing to my forward right*). This limits the agent’s representational power in favor of focusing attention on a small number of important outcomes.

an agent learns a nonlinear function approximator that provides a mapping from the atomic state IDs to action-value (“ $Q$  function”) estimates. In this paper we are concerned with the latter approach, in which the agent is provided with a *relational language* – a set of logical predicates and constants that describes the state space in terms of objects, their properties, and relationships among them. This language provides an abstraction between atomic states and the agent’s action-value function, allowing the agent to generalize across classes of related states.

In the relational approach, unary predicates describe *properties* of objects, while  $n$ -ary predicates describe *relationships* among objects. If the state conjunct is incomplete (either because it does not contain enough terms to uniquely identify the state or because some constants are replaced by variables) then the state representation is *abstract* – it refers to a *set* of atomic states. Formally, these atomic states are aliased and the agent assigns the same action-value to all members of the set. When these states are, in fact, value-equivalent, this aliasing is advantageous and allows the agent to apply experience from a single atomic state/action pair to all elements of the relational set. (Sufficient conditions for this equivalence are developed by algebraic equivalence theories for MDPs [9,10,11], which provide theoretical justification for this approach.) When the atomic states in the relational set are *not* value-equivalent, however, disaster can ensue – the agent’s policy may oscillate, diverge, or converge to an incorrect value, because it is trying to assign one policy when more than one is needed.

The difficulty, then, is ensuring that the relational representation captures enough of the state description to function effectively, while discarding enough to generalize well. The approach we adopt here is based on state *envelopes*: the agent maintains an explicit representation of some set of states “near” it (the envelope) and disregards anything beyond that set [12,13,14]. The intuition is that the agent can (often) act well with only local information, so it need not spend time or memory planning on distant states that it is unlikely to encounter. Traditionally, the envelope is seen as a subset of the state space, and the decision-theoretic problem has been deciding which states should be kept within the envelope. In this work, we view the envelope as a description of the local space around the agent, encoding the locations that it is likely to encounter within only a few steps. We can do this by virtue of a *metric* on the environment that allows the agent to analytically determine reachable locations without exploration. Rather than dynamically expanding the envelope over time, we instead keep a fixed envelope, centered on the agent. The envelope moves with the agent, adding new elements and discarding old as the agent moves through the environment. The intuition here is that local information is often sufficient for navigation. By remembering previous experience with isomorphic envelopes, the agent can generalize experience across local problems. An example of this is: “There is a tree directly in front of me. I already know from a similar task that I should not attempt to go through this tree – I will instead go around.” There are, of course, environments in which rich global information is critical to navigation, but we focus here on tasks in which only limited global information (direction

to the goal) is necessary for strong performance. For example, the hiker in the forest may need to know only the direction to the campsite in order to reach it.

This representation is close to deictic in that the agent is reasoning in terms of “the terrain right in front of me; the terrain to my right; the terrain two units to my front-left;” and so on. For a fixed envelope size, this representation is formally less powerful than a full relational state description (e.g., it can be propositionalized), but a relational language provides a convenient mechanism for abstracting away from atomic state IDs. The local knowledge aspects of this representation are also similar to chart-based topological RL methods [15].

### 3 Navigational MDPs and Envelopes

In this paper, we restrict our attention to (discrete) spatial navigation tasks. First, we will describe the full MDP in which the agent is executing and the corresponding atomic-state representation. Then we will describe the envelope-based representation that the relational agent uses.

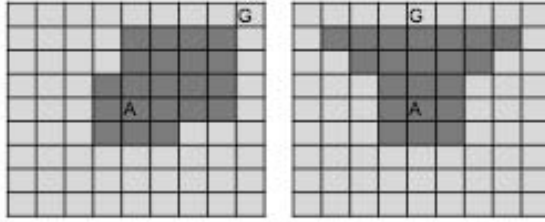
Both agents use the same action space:  $\mathcal{A} = \{\text{FWD}, \text{TURN}_L, \text{TURN}_R\}$ , denoting “move forward”, “turn counter-clockwise”, and “turn clockwise”, respectively. The agents move through a world consisting of a set of *locations*,  $\mathcal{L} \subseteq \mathbb{Z} \times \mathbb{Z}$ , specified via Cartesian coordinates. Each location is assigned a *terrain type* property; for example,  $\text{WALL}(l)$  or  $\text{GRASS}(l)$  denotes that location  $l \in \mathcal{L}$  is a wall segment or a patch of grass, respectively. The complete set of terrain type properties is denoted  $\mathcal{P}$ . Locations are laid out on a square grid and each location is reachable from its eight neighbors. The agent’s transition function depends on the terrain type of the locations it is moving from and to and on other surrounding terrain. The complete set of terrain types we use in our experiments, and their corresponding dynamics, are given in Section 3.1. Note that transition dynamics depend only on the *properties* of locations and not on the absolute coordinates of locations. Altogether, the set of terrain types for all locations in  $\mathcal{L}$  is the *terrain map*,  $M : \mathcal{L} \rightarrow \mathcal{P}$ .

Further, the environment is endowed with a *metric* given by the relation  $d(l_i, l_j)$ , which specifies distances between locations, and a *direction*,  $\phi(l_i, l_j)$ , which gives orientation between them (with respect to an arbitrary fixed reference direction). Together,  $d$  and  $\phi$  establish a coordinate frame and provide a set of relations that allow the agent to reason about “the location immediately to my right” or “the location 10 units to my northeast.” In the experiments reported here, we use a Manhattan distance  $d$ , with respect to the 8-neighbor connectivity, and a discretized angle  $\phi$ ,  $\Phi = \{\text{NORTH}, \text{NORTHWEST}, \text{WEST}, \dots\}$ , but other distance/angle relations are possible. At any instant, the agent’s (atomic) state,  $s_{\text{atom}}$ , is given by its current location,  $l_a \in \mathcal{L}$ , and its facing,  $f_a \in \Phi$ ; the set of all such pairs is the atomic state space,  $\mathcal{S}_{\text{atom}} = \mathcal{L} \times \Phi$ .

We say that an *environment* is the tuple  $E = \langle \mathcal{L}, \mathcal{P}, M, d, \phi \rangle$ . Together with the transition dynamics for the terrain types,  $T_{\mathcal{P}}$ , the environment captures the transition function of a Markov decision process. In addition, there is a distinguished *goal location*,  $l_g \in \mathcal{L}$ . The reward function,  $R$ , is 1 at the goal

location (regardless of facing) and 0 elsewhere. The full MDP, then, is given by  $\mathcal{M} = \langle \mathcal{S}_{\text{atom}}, \mathcal{A}, T_{\mathcal{P}}(M), R \rangle$ . An agent that uses atomic states for its representation is executing directly in  $\mathcal{M}$  and, in principle, can learn a stationary, deterministic policy for that environment [16].

Unfortunately, the atomic state representation does not handle nonstationary MDPs well – absolute coordinates and directions become meaningless if the goal location or environment changes. Thus, the *relational agent* uses an *envelope* representation that captures only the properties of locations “near” the agent, as well as some information about the (relative) location of the goal.



**Fig. 1.** Two envelopes, each with an envelope radius of 3. The shaded area represents the envelope, while  $A$  represents the agent, and  $G$  represents the goal. The envelopes are different shapes because the agent must be able to reach every location in the envelope in 3 steps or less.

For an agent at location  $l_a$ , the agent-centric envelope about  $l_a$  is  $e(l_a) \subset \mathcal{L}$ . A simple  $e(l_a)$  would be a disk of radius  $r$ :  $e(l_a) = \{l_i \in \mathcal{L} : d(l_a, l_i) \leq r\}$ . We have found, however, that a more complex envelope geometry (Fig. 1) is more effective in these environments. This configuration is roughly a quarter-disk arc of radius  $r$  pointing toward the goal, plus the locations immediately adjacent to the agent. The agent can perceive the terrain type property of all locations in its envelope as an ordered tuple,  $P(e(l_a)) = (M(l_i))_{l_i \in e(l_a)}$ , but cannot directly perceive the location coordinates themselves. Thus, the envelope representation is independent of absolute location. This is an attempt to exploit “translation invariance” – a special case of MDP homeomorphism [11]. Translation invariance, in turn, depends on homogeneity of the transition function (provided by the terrain type properties) and invariance of the metric  $d$ .

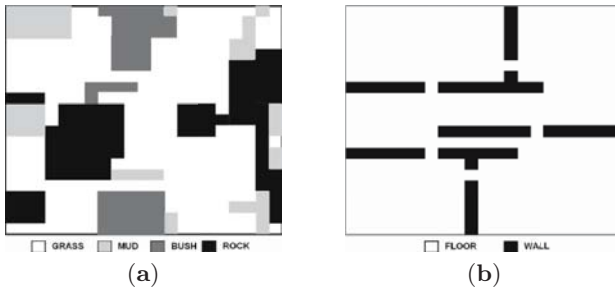
The relational agent also perceives the direction and the distance to the goal. The distance is quantized such that for some fixed distance,  $d_{\text{quant}}$ , if  $d(l_a, l_g) \leq d_{\text{quant}}$  then the agent perceives  $\hat{d} = d(l_a, l_g)$ . However, for all  $d(l_a, l_g) > d_{\text{quant}}$ , the agent perceives only the symbol  $\hat{d} = \text{FAR AWAY}$ . The direction and quantized distance provides a small amount of global information and helps alleviate some aliasing difficulties by allowing the agent to make different decisions depending on the relative position of the goal. The relational agent’s state representation, then, is  $s_{\text{rel}}(l_a) = \langle P(e(l_a)), \phi(l_a, l_g), \hat{d}, f_a \rangle$ , and the set of all such state representations is  $\mathcal{S}_{\text{rel}}$ .

Note that  $s_{\text{rel}}$  contains some information about  $l_g$ , while  $s_{\text{atom}}$  does not. This is analogous to our hiker in the woods knowing “I need to head roughly North and my destination is still too far to see.” Once the hiker draws close to the goal, it becomes visible. This representation “cheats” a bit by giving some information to the relational agent that is unavailable to the atomic agent. In Section 4.2 we demonstrate that the relational agent still has an advantage, even after controlling for this extra information.

### 3.1 Agent Dynamics

Our agent dynamics roughly simulate a “noisy” robot that has to cope with varied terrains and some global structure.

An agent learns the task of finding a single goal in simulated, stochastic, 21x21 indoor and outdoor environments. Indoor environments consist of FLOOR and WALL terrain types, with fixed WALL elements at the boundaries so the agent cannot move beyond the edges of the environment. Outdoor worlds consist of GRASS, BUSH, MUD, and ROCK terrain types with a toroidal topology; when an agent attempts to move past a boundary it appears on the opposite edge of the environment. The environments are randomly generated according to a *terrain generation distribution* designed to produce roughly realistic terrain maps. The indoor terrain generator produces room-like structures and doorways (guaranteed to prevent unreachable locations); the outdoor terrain generator distributes blocks of each terrain type so as to create varied regions of terrain. Examples of each environment type are shown in Fig. 2.



**Fig. 2.** Examples of randomly generated (a) outdoor and (b) indoor environments

Each terrain type has several attributes determining the outcome of an attempted move into or within that terrain. These attributes are summarized in Table 1.

The **enter** attribute is the probability that the FWD action will be successful – that the agent will enter the target location of the FWD action. That is,  $\text{Pr}[\text{enter}|\text{GRASS}]$  is the probability that the agent will move into the GRASS cell that it is facing when it invokes the FWD action. When FWD fails, the agent might

**Table 1.** Transition dynamics for  $\mathcal{P}$ . Each cell of the table gives  $\Pr[\text{Outcome}|\text{Terrain}]$ .

	Target Terrain Type				
	FLOOR	WALL ROCK	GRASS	BUSH	MUD
<b>enter</b>	0.95	0.0	0.85	0.3	0.9
<b>slideRight</b>	0.05	0.15	0.1	0.2	0.25
<b>slideLeft</b>	0.05	0.15	0.1	0.2	0.25
<b>moveOvershoot</b>	0.05	0.0	0.05	0.001	0.2
<b>turnRight</b>	0.95	0.0	0.9	0.4	0.7
<b>turnLeft</b>	0.95	0.0	0.9	0.4	0.7
<b>turnOvershoot</b>	0.05	0.0	0.05	0.001	0.35

**slideRight** or **slideLeft** of the target location, with probability given by the respective attribute of the target location’s terrain type. The result of **slideRight** is an attempt to **enter** the location to the right (with respect to the agent’s current facing) of the target location (which may, itself, result in a **slideRight**, etc.). When the FWD action succeeds, the agent may **moveOvershoot** the target location, attempting to **enter** the next location in its direction of movement. The agent changes its facing with the  $\text{TURN}_R$  ( $\text{TURN}_L$ ) action, which succeeds with probability **turnRight** (**turnLeft**) of the terrain type of the agent’s current location. When the agent successfully turns, there is some chance of a **turnOvershoot**, leading to a successive **turn\*** attempt. Any probability mass not otherwise accounted for leaves the agent’s state unchanged.

### 3.2 Learning Algorithm and Parameters

Both agents use a standard tabular  $Q$ -learning algorithm [17]; the only difference between them is their state representation. The atomic agent learns the state-action value function  $Q : \mathcal{S}_{\text{atom}} \times \mathcal{A} \rightarrow \mathbb{R}$ , while the relational agent learns the function  $Q : \mathcal{S}_{\text{rel}} \times \mathcal{A} \rightarrow \mathbb{R}$ .

We set the  $Q$ -learning parameters,  $\alpha$ ,  $\gamma$ , and  $\epsilon$ , empirically from initial experiments in indoor environments. We found that the values  $\alpha = 0.1$ ,  $\gamma = 0.7$ , and  $\epsilon = 0.01$ , while not universally optimal, appear to work adequately in all tested environments. The relational state representations use an  $r = 3$  and a  $d_{\text{quant}} = 6$ , which performed well in simple initial indoor and outdoor tests. Performance does not appear to be strongly sensitive to these parameters but, for consistency, we use these fixed values in all of our experiments.

## 4 Experiments

Our experiments are designed to demonstrate the adaptation capacity of the relational agent in the face of two kinds of nonstationarity: changes in the location of the goal ( $l_g$ ) and changes in the environment ( $E$ ).

We have experimented with fixed goals, fixed environments, random goals, and random environments. In all configurations a *trial* runs until the agent reaches

$l_g$ , at which point the agent is rewarded and relocated to a random starting location in  $\mathcal{L}$ , and a new trial begins. If a single trial reaches 10,000 steps, it is truncated and the agent is restarted.

In a *fixed environment* (FE) experiment,  $E$  does not change throughout the experiment. In a *random environment* (RE) experiment,  $E$  changes every 10 trials by drawing  $M$  from a set of 100 pre-generated maps (which are, in turn, drawn according to the appropriate terrain generation distribution – Section 3.1). In a *fixed goal* (FG) experiment,  $l_g$  is held unchanged at the center of  $\mathcal{L}$  throughout the experiment. In a *random goal* (RG) experiment,  $l_g$  is drawn uniformly at random from  $\mathcal{L}$  after each trial. The pseudorandom number generators are synchronized for the atomic and relational agents, so they both experience the same sequences of start locations, goal locations, and environments.

In all experiments, the agent is trained for a total of 200,000 trials. Every 100 trials, its policy is frozen and its performance is evaluated for 50 trials; we report mean performance for the 50 evaluation trials.

#### 4.1 Nonstationary Goal Locations

We first examined the agents’ robustness to changes in  $l_g$  with a FE, RG experiment. Results for both agent types in indoor and outdoor environments are shown in Fig. 3 (a) and (b).

The agent with the atomic state representation does poorly in both environments, while the envelope-based relational agent exhibits much stronger performance. The relational agent converges to a stable policy with low performance variance within roughly 60,000 trials, while the atomic agent never converges and continues to have wildly varying performance for the entire 200,000 trials.

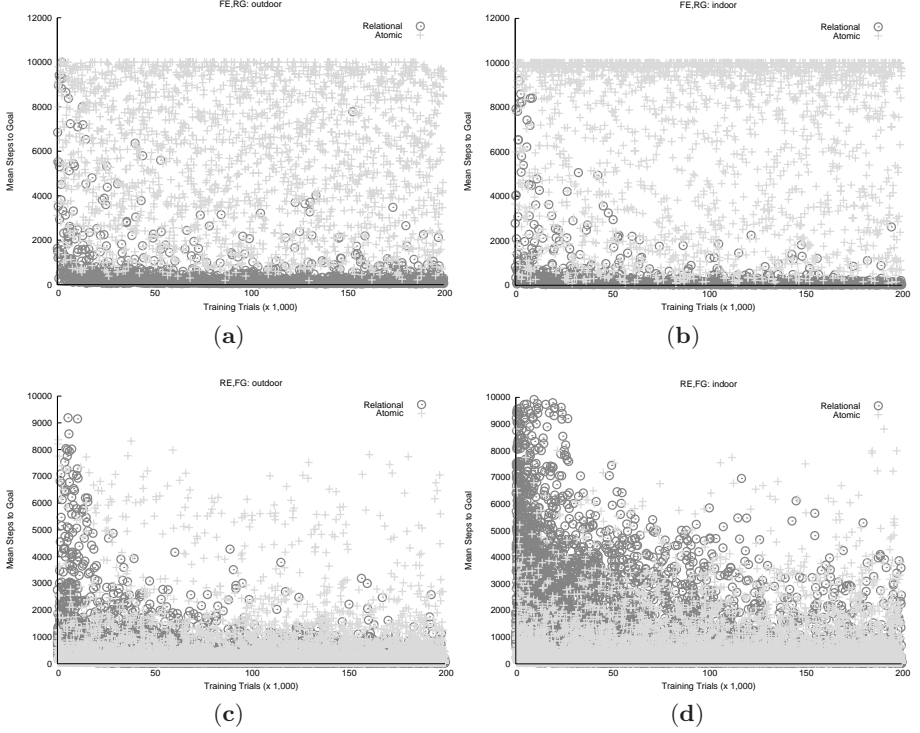
These findings are, perhaps, not surprising –  $Q$ -learning is guaranteed to converge only for a stationary MDP. By changing  $l_g$ , the atomic agent’s value function gradients become essentially meaningless – its only long-term strategy can be a random walk. The relational agent, on the other hand, knows the *relative* location of  $l_g$  and can build stable value functions whose gradients always indicate the direction to the goal. In spite of the partial observability imposed by the envelope, the relational representation maps the nonstationary MDP into a (nearly) stationary representation, allowing the policy to converge.

#### 4.2 Nonstationary Environments

We next studied the agents’ robustness to changes in  $E$  with a RE, FG experiment. Note that fixing the goal location across all terrain maps controls for the extra “location of goal” information that the relational agent possesses, because the goal is at the same atomic state in all trials. Thus, any performance differences here are because the relational agent is better able to adapt to changing environments and not because it has an informational advantage over the atomic agent. Results for both agent types in outdoor and indoor environments are shown in Fig. 3 (c) and (d).

The relational agent finds a stable policy in both indoor and outdoor worlds, though convergence is slower here than in the FE, RG scenario. The atomic agent





**Fig. 3.** Learning performance for both atomic and relational agents. (a) and (b) display fixed-environment, random-goal results (FE, RG); (c) and (d) give random-environment, fixed-goal (RE, FG). Performance in outdoor environments is given in (a) and (c), while indoor environments are shown in (b) and (d). The horizontal axes of all plots give number of trials, while the vertical axes show per-trial steps-to-goal, averaged over policy evaluation trials. Circles denote relational agent performance; pluses show atomic performance. Note that trials were truncated at 10,000 steps if the agent had not yet located  $l_g$ .

performs substantially better here, finding a reasonable policy, but it never converges to performance as strong as relational’s. Further, its performance continues to be high-variance for the duration of the experiment, while relational’s variance declines over time. Over the last 25,000 trials, the atomic agent’s mean performance is 216.5 steps to goal in the outdoor environments and 323.2 in the indoor (standard deviation of 557.8 and 732.2, respectively). The relational agent, on the other hand, achieves means of 77.6 and 125.4 steps in outdoor and indoor environments (standard deviation of 117.0 and 364.3, respectively).

These results show that the relational agent is successfully accumulating knowledge about the different worlds over time. More importantly, the relational agent is essentially learning to quickly *recognize* which environment it is currently running in and is executing a reasonable policy for that environment.



This is fairly strong performance, given that this scenario is equivalent to a single POMDP consisting of  $100 \cdot |\mathcal{L} \times \Phi| = 352,800$  states, including a single hidden variable (current environment).

More surprising is how well the atomic agent *does* fare in this case. As we argued in Section 4.1, we might expect nonstationarity to upset  $Q$ -learning in general, and we saw that goal nonstationarity did overwhelm the atomic agent. Here, however, the goal location always maps to the same absolute  $l_g$ , so absolute coordinate value gradients remain (roughly) consistent over time. Still, if the environment changed radically enough, it should still disrupt the atomic agent (imagine a series of complex mazes in which the correct action at any location changes on every change of  $M$ ).

It appears that environmental homogeneity arising from the terrain generation distribution is helping the atomic agent here. The best direction to the goal is likely to remain roughly the same across many environments, so the optimal value function gradients are approximately stable. However, the atomic agent has no real knowledge of the local structure of the environment – when a draw from the terrain generator produces significantly different terrain, requiring different local navigation policies, the atomic agent experiences a setback and must re-learn that environment. The relational agent, on the other hand, can often find an envelope that provides precedent information on how to handle this terrain. Thus, in the long-run, the relational agent is learning about the terrain generation distribution itself and achieving a more stable policy, while the atomic agent is successful only to the extent that the terrain generator tends to produce similar terrain in the same absolute states.

It also appears that indoor environments are more difficult than outdoor – both agents reach worse mean and standard deviation performance in indoor than out. We believe that this is a function of the global topology of the environment. In the outdoor worlds, “head toward the center of the map” is usually the roughly correct policy. In indoor worlds, however, there may be room configurations that force the agent to go “the long way around” to get to the goal. Such configurations are beyond the agent’s perception (envelope) so it cannot learn to do the right thing. Still, the relational agent has learned to do fairly well given only local information.

We have also experimented with RE, RG scenarios and found the results to be consistent with the FE, RG scenario, indicating that the changes in goal states dominate changes in environment for the atomic agent.

## 5 Conclusion

Overall, we believe that the success of the envelope-based navigation is due to the spatial nature of these environments. Unlike a blocks-world domain, there is a strong metric here that allows the agent to define an envelope of locations that it is likely to encounter in the next few steps. Indeed, the shape of the envelopes we use (Fig. 1) excludes locations that are unlikely to be encountered in the short term. Thus, the partial observability of the relational representation penalizes us

far less than the deictic representation hurt Finney *et al.*'s blocks-world agents [2002].

We had hoped to demonstrate that the relational agent could use its knowledge of the terrain generation distribution to *generalize* to novel (previously unexperienced) environments. It turns out to be difficult to experience enough distinct envelope configurations to achieve strong generalization performance, however – our agents began to overflow system memory before achieving convincing generalization. Still, few, if any, other RL algorithms have demonstrated successful control learning in rapidly changing sets of environments. This type of control learning might be useful to an agent that needs to function in any of a set of different buildings or different cities. We believe that the addition of function approximation will allow the relational agent to achieve task generalization. We also intend to exploit additional symmetries and invariances (e.g., rotational) in future implementations.

We are also interested in the local vs. global topology issue that we observed in Section 4.2. These results reflect the tradeoff between purely local/reactive control and global/planning-based control that has been debated for years in robotics. We believe that a hybrid method, that exploits both local, RL-based navigation and global, planning-based navigation will be successful here, but such an approach remains future work for the moment.

## Acknowledgements

Thanks to Patrick Carlson and Laura Glendenning for brainstorming and proof-reading on this work. Thanks to our reviewers for valuable feedback and suggestions. This material is based upon work supported by the National Science Foundation under Grant No. 0453545.

## References

1. Finney, S., Gardiol, N.H., Kaelbling, L.P., Oates, T.: The thing that we tried didn't work very well: Deictic representation in reinforcement learning. In: UAI-2002. (2002)
2. McCallum, A.: Overcoming incomplete perception with utile distinction memory. In: ICML-93. (1993)
3. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Optimization and neural computation series. Athena Scientific, Belmont, MA (1996)
4. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research 4(December) (2003) 1107–1149
5. Mahadevan, S.: Proto-value functions: Developmental reinforcement learning. In: ICML-2005. (2005)
6. Džeroski, S., De Raedt, L., Driessens, K.: Relational reinforcement learning. Machine Learning 43(1–2) (April 2001) 7–52
7. van Otterlo, M.: A survey of reinforcement learning in relational domains. Technical Report TR-CTIT-05-31, University of Twente, Centre for Telematics and Information Technology (July 2005) ISSN 1381-3625.

8. Fern, A., Yoon, S., Givan, R.: Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *Journal of Artificial Intelligence Research* **25** (2006) 75–118
9. Dean, T., Givan, R.: Model minimization in Markov decision processes. In: *AAAI-97*. (1997) 106–111
10. Ravindran, B., Barto, A.G.: Relativized options: Choosing the right transformation. In: *ICML-2003*. (2003) 608–615
11. Ravindran, B.: An Algebraic Approach to Abstraction in Reinforcement Learning. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, MA (2004)
12. Tash, J., Russell, S.: Control strategies for a stochastic planner. In: *AAAI-94*. (1994)
13. Dean, T., Kaelbling, L.P., Kirman, J., Nicholson, A.: Planning under time constraints in stochastic domains. *Artificial Intelligence* **76** (1995)
14. Baum, J., Nicholson, A.E.: Dynamic non-uniform abstractions for approximate planning in large structured stochastic domains. Technical Report 1998/18, School of Computer Science and Software Engineering, Monash University, Melbourne (1998)
15. Glaubius, R., Smart, W.D.: Manifold representations for value-function approximation in reinforcement learning. Technical Report 05-19, Department of Computer Science and Engineering, Washington University in St. Louis (2005)
16. Puterman, M.L.: *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York (1994)
17. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA (1998)

# On the Stability and Bias-Variance Analysis of Kernel Matrix Learning

V. Vijaya Saradhi and Harish Karnick

Dept. of Computer Science and Engineering  
Indian Institute of Technology Kanpur  
India  
{saradhi,hk}@cse.iitk.ac.in

**Abstract.** Stability and bias-variance analysis are two powerful tools to understand learning algorithms better. We use these tools to analyze learning the kernel matrix (LKM) algorithm. The motivation comes from: (i) LKM works in the transductive setting where both training and test data points are to be given apriori. Hence, it is worth knowing the stability of LKM under small variations in the data set and (ii) It has been argued that LKMs overfit the given data set. In particular we are interested in answering the following questions: (a) *Is LKM a stable algorithm?* (b) *do they overfit* (c) *what is the bias behavior with different optimal kernels?*. Our experimental results show that LKMs do not overfit the given data set. The stability analysis reveals that LKMs are *unstable* algorithms.

**Keywords:** Kernel matrix learning, stability, bias-variance, error estimation, bootstrap, overfitting.

## 1 Introduction

A support vector machine (SVM) is a powerful learning algorithm for binary classification tasks. It is based on the principle of structural risk minimization [3,5,11]. SVMs attracted researchers due to their promising generalization performance [11]. This has lead to much work on error analysis for SVMs. The leave-one-out (LOO) technique is the most widely used for error estimation [12], [13]. Stability analysis for SVMs has been carried out by Buciú *et al.*, [2] and they show experimentally that SVMs are stable. That is bagging [1] based strategies do not improve performance of SVMs. For a learning algorithm, if small changes in the training data set lead to large variations in the performance of the *learnt classifier*, then the algorithm is said to be *unstable* otherwise it is said to be stable.

Bias-variance analysis has been used to study a learning algorithm and it is useful in understanding relationships between the free parameters *viz.*,  $C$  and  $\sigma$  on error, bias and variance. An experimental study of the bias-variance has been carried out by [10] for SVMs. The experimentation has thrown light on several questions; in particular how to use bias-variance analysis for developing

ensemble methods in which SVMs are one of the base learners, predicting low error region using variation in number of support vectors and relation between bias, variance and VC dimension. The key idea in both stability and bias-variance analysis is to decompose the error into bias and variance terms and analyze them independently.

These tools are also useful in the analysis of focused methods aimed at improving the performance of SVMs. In particular the parameter tuning/LKM algorithm [4,7]. In the present work, we analyze LKM empirically using the tools *viz.*, stability and bias-variance. Motivation for analyzing the LKM comes from two directions (i) LKM works in the transductive setting where both training and test data points are to be given apriori; this is of particular importance as LKM depend on test data points as well and (ii) It has been argued that LKMs overfit the given data set [8].

The performance of SVMs depends completely on the kernel function choice. Therefore choosing a *good* kernel function is at the heart of improving SVMs performance on test data points. The best choice of a kernel function for a given problem has been addressed very recently in [7]. The central idea in this is to express the kernel matrix as a weighted linear combination of a set of a priori chosen kernels ( $K = \sum_{i=1}^m \mu_i K_i$ ) which are positive semidefinite with a bound on the trace of the resulting kernel matrix. The objective is to learn the optimal kernel matrix by learning weights associated with individual kernel matrices. This optimization problem is posed as a semidefinite programming problem (SDP).

We have conducted experiments on 5 real world and artificial data sets. Experimentally we observe the following: (i) LKMs do not overfit. This contradicts what has been said in the literature about LKMs [8] and (ii) they are unstable algorithms.

This paper is organized as follows: learning the kernel matrix (LKM) algorithm is presented in the next section. Bootstrap re-sampling and error estimation using bootstrap is discussed in section 3. Section 4 discusses stability of a learning algorithm. Bias-variance, error decomposition is presented in section 5. Experimental results of stability and bias-variance for the LKM are presented in section 6. We conclude with a summary.

## 2 Learning the Kernel Matrix

In [7], the optimal kernel matrix is expressed as a weighted linear combination of kernels chosen from an a priori set  $\mathcal{K} = \{K_1, K_2, \dots, K_m\}$  (which is a convex subset of positive semidefinite matrices) with bounded trace. That is: (1)  $K = \sum_{i=1}^m \mu_i K_i$  (2)  $K \succeq 0$  (3)  $\text{trace}(K) \leq c$ . Note that kernel matrix learning is done in the transductive setting wherein the kernel matrix has entries corresponding to both training as well as test data points. The aim is to find optimal  $\mu_i$ s. This in turn is achieved by solving the following semidefinite programming (SDP) problem [7].

**Convex Optimization Formulation 1.** *Given a training set  $\mathcal{X}$ , the optimum kernel matrix  $K$  can be found by solving the following convex optimization problem:*

$$\begin{aligned} & \min_{\mu, \lambda, \nu, \delta} t \\ & \text{subject to } \text{trace}(\sum_{i=1}^m \mu_i K_i) = c \\ & \quad \sum_{i=1}^m \mu_i K_i \succeq 0 \\ & \quad \begin{pmatrix} G(\sum_{i=1}^m \mu_i K_{i, tr}) & (\mathbf{e} + \nu - \delta + \lambda \mathbf{y}) \\ (\mathbf{e} + \nu - \delta + \lambda \mathbf{y})^T & t - 2C\delta^T \mathbf{e} \end{pmatrix} \succeq 0 \\ & \quad \nu \succeq 0, \delta \succeq 0 \end{aligned} \quad (1)$$

Minimizing  $t$  is equivalent to maximizing the dual formulation of SVM.  $\mathbf{e}$  is a vector of all ones.  $G(K)$  is defined as  $G_{ij}(K) = y_i y_j k(\mathbf{X}_i, \mathbf{X}_j)$ .  $\nu, \delta, \lambda$  are the Lagrangian multipliers associated with the constraints.  $X \succeq 0$  denotes that matrix  $X$  should be positive semidefinite. The optimal kernel matrix  $K$  is obtained using the optimal weights  $\mu_i^*$ s by solving equation (1). For classifying a test data point  $\mathbf{X}$ , the following equation is used:  $y = f(x) = \text{sign} \left( \sum_{i=1}^m \sum_{j=1}^N \mu_i^* \alpha_j y_j k_i(\mathbf{X}_j, \mathbf{X}) \right)$ .

### 3 Bootstrap Error Estimation

Both stability and bias-variance analysis requires *perturbed* data set generated from the original training data set. This is achieved through the *Bootstrap*, a well known re-sampling technique in the statistical literature. In this re-sampling technique,  $B$  independent data sets *viz.*,  $T_1, T_2, \dots, T_B$  each of size  $N$  are drawn from the original training data set  $\mathcal{X} = \{(\mathbf{X}_i, y_i)\}_{i=1}^N$ . From these independent data sets, test data points for  $b^{th}$  realization are obtained as  $T_b^{tst} = \mathcal{X} - T_b$ . Let  $I_i^b$  be an indicator function denoting whether  $i^{th}$  data point in  $b^{th}$  bootstrap realization is a test point or not.

$$I_i^b = \begin{cases} 1 & \text{if } \mathbf{X}_i \in T_b^{tst} \\ 0 & \text{Otherwise} \end{cases}$$

The estimated generalization error is computed using the leave-one-out bootstrap as follows [2]:

$$\widehat{err}(C) = \frac{1}{N} \sum_{i=1}^N \frac{\sum_{b=1}^B I_i^b \ell(y_i, f_i^b)}{\sum_{b=1}^B I_i^b} \quad (2)$$

where  $\ell(a, b)$  is a 0/1 loss function. It assumes a value 1 if  $a \neq b$  and a value 0 if  $a = b$ .  $f^b$  is a decision function built using  $b^{th}$  bootstrap realization and  $f_i^b$  stands for the decision given by  $f^b$  on a data point  $\mathbf{X}_i$ .

### 4 Stability Analysis

The stability of a learning algorithm is quantified in terms of ratio of the average prediction error (equation 2) and average aggregate error [2]. For the  $i^{th}$  data

point, the aggregate error is the decision obtained by considering a voting scheme on all the  $B$  classifiers built using bootstrap realizations. The aggregate error is computed as follows: Construct classifiers,  $f^b$ , for all bootstrap realizations  $b = 1, \dots, B$ . For classifying  $i^{th}(\mathbf{X}_i)$  data point, compute  $f_i^b$  (that is  $f^b(\mathbf{X}_i)$ ) for all  $b = 1, \dots, B$  and a decision is made based on voting. For voting, we count votes only when  $\mathbf{X}_i \in T_b^{tst}$ ,  $\forall b = 1, \dots, B$ . This is given in equation 3.

$$\text{sign} \left\{ \frac{\sum_{b=1}^B I_i^b f_i^b}{\sum_{b=1}^B I_i^b} \right\} \quad (3)$$

The average aggregate error is computed over all data points and is given in equation 4 [2].

$$\widehat{err}(C_a) = \frac{1}{N} \sum_{i=1}^N \left\{ \frac{\sum_{b_1=1}^B I_i^{b_1} \ell \left( y_i, \text{sign} \left\{ \frac{\sum_{b_2=1}^B I_i^{b_2} f_i^{b_2}}{\sum_{b_2=1}^B I_i^{b_2}} \right\} \right)}{\sum_{b_1=1}^B I_i^{b_1}} \right\} \quad (4)$$

An algorithm is said to be stable if the stability factor  $\hat{\delta} \leq 1$  and is given by:

$$\hat{\delta} = \frac{\widehat{err}(C)}{\widehat{err}(C_a)} \leq 1. \quad (5)$$

Otherwise the algorithm is considered to be unstable. Equation 5 tells us that if  $\hat{\delta} > 1$ , then  $\widehat{err}(C_a) < \widehat{err}(C)$ . This means that for small perturbations in the training data (bootstrap realizations in the present context), bagging improves the performance leading to instability. Using this notion, [2] has shown SVMs to be stable algorithms experimentally.

## 5 Bias-Variance Analysis

Giorgio *et al.*, [10] decomposed the error into bias and variance. Further, variance is divided into two components (i) biased-variance and (ii) unbiased-variance. We use the definitions given in [10] for the above terms. Assuming a two-class classification problem, bias for data point  $\mathbf{X}_i$ , denoted as  $B(\mathbf{X}_i)$ , is a loss function between main prediction ( $y_m$ ) and the true label and is given as:

$$B(\mathbf{X}_i) = \ell(y_m, y_i) \quad (6)$$

where  $y_m = \arg \max(p_1, p_{-1})$  is the main prediction and

$$p_1 = \frac{1}{\sum_{b=1}^B I_b^i} \sum_{b=1}^B ||I_i^b \text{ and } f_i^b = 1|| \quad (7)$$

$$p_{-1} = \frac{1}{\sum_{b=1}^B I_b^i} \sum_{b=1}^B ||I_i^b \text{ and } f_i^b = -1|| \quad (8)$$

Variance of  $\mathbf{X}_i$  is the variation in decisions across all  $B$  classifiers relative to the main prediction and is computed as:

$$V(\mathbf{X}_i) = \frac{1}{\sum_{b=1}^B I_b^i} \sum_{b=1}^B ||I_i^b \text{ and } y_m \neq f_i^b|| \quad (9)$$

Un-biased variance and biased variance are computed as given below:

$$V_u(\mathbf{X}_i) = \frac{1}{\sum_{b=1}^B I_b^i} \sum_{b=1}^B ||I_i^b \text{ and } y_m \neq f_i^b \text{ and } B(\mathbf{X}_i) = 0|| \quad (10)$$

$$V_b(\mathbf{X}_i) = \frac{1}{\sum_{b=1}^B I_b^i} \sum_{b=1}^B ||I_i^b \text{ and } y_m \neq f_i^b \text{ and } B(\mathbf{X}_i) = 1|| \quad (11)$$

Net variance is the difference between unbiased variance and biased variance. That is  $V_n(\mathbf{X}_i) = V_u(\mathbf{X}_i) - V_b(\mathbf{X}_i)$ .

In the bias-variance analysis, the behavior of these five quantities (bias, variance, unbiased variance, biased variance and net variance) are characterized with respect to the free parameters in the LKM.

## 6 Experimental Results

In our experiments, we considered four real world data sets namely **heart**, **titanic**, **thyroid** and **waveform** [9] and an artificial data set, P2 [10]. Characteristics of each data set are given in table 1. We considered the first 100 test data points for titanic and waveform data. We solve equation (1) using self dual minimization package **SeDuMi** [6]. For all the data sets, 100 independent realizations were generated (that is  $B = 100$ ). We have experimented with the optimal kernel matrix composed of (1) only Gaussian functions (2) only polynomial functions and (3) mixture of Gaussian and polynomial functions. The free parameter,  $C$ , takes the values (0.01, 0.1, 1.0, 10.0 and 100.0). We have solved 7500 SDPs ( $3 \times 5 \times 100 \times 5 = 7500$ ; 3 different optimal kernel types, 5 different values for  $C$ , 100 independent realizations and 5 data sets) in total and report results based on these experiments for both stability analysis and bias-variance analysis.

In the case of stability analysis, we have computed the average prediction error and average aggregation error ( $\widehat{err}(C)$  and  $\widehat{err}(C_a)$ ) and the stability factor  $\hat{\delta}$ . We analyse the relationship between average error, bias, unbiased-variance, biased-variance, number of support vectors, training error and the user parameter  $C$ . The following combination of the optimal kernels were considered in our experiments:

1. **Only Gaussian kernels:** We have considered three Gaussian kernel functions with varying spread widths such that one of the spread widths ( $\sigma_1$ ) is the optimal parameter for SVMs and rest of the spread widths are ten times the optimal value and  $\frac{1}{10}^{th}$  of the optimal value.



**Table 1.** Data sets considered for experimentation

Data Set	# features	# of trg. samples	# of test samples
P2	2	100	100
heart	13	170	100
thyroid	5	140	75
titanic	3	150	2050
waveform	21	400	4600

2. **Only polynomial kernels:** In this case, we have considered three polynomial kernel functions with varying degrees ( $d_1 = 3, d_2 = 5$  and  $d_3 = 7$ ).
3. **Mixture of Gaussian and Polynomial kernels:** We considered two Gaussian kernel functions and one polynomial kernel function. The  $\sigma$ 's in the case of Gaussian kernel function are the optimal  $\sigma_1$  obtained for SVMs for the data set under consideration and  $\sigma_2 = 10 \times \sigma_1$ . For polynomial kernel function, we use degree as  $d_1 = 3$ .

### 6.1 Stability Analysis

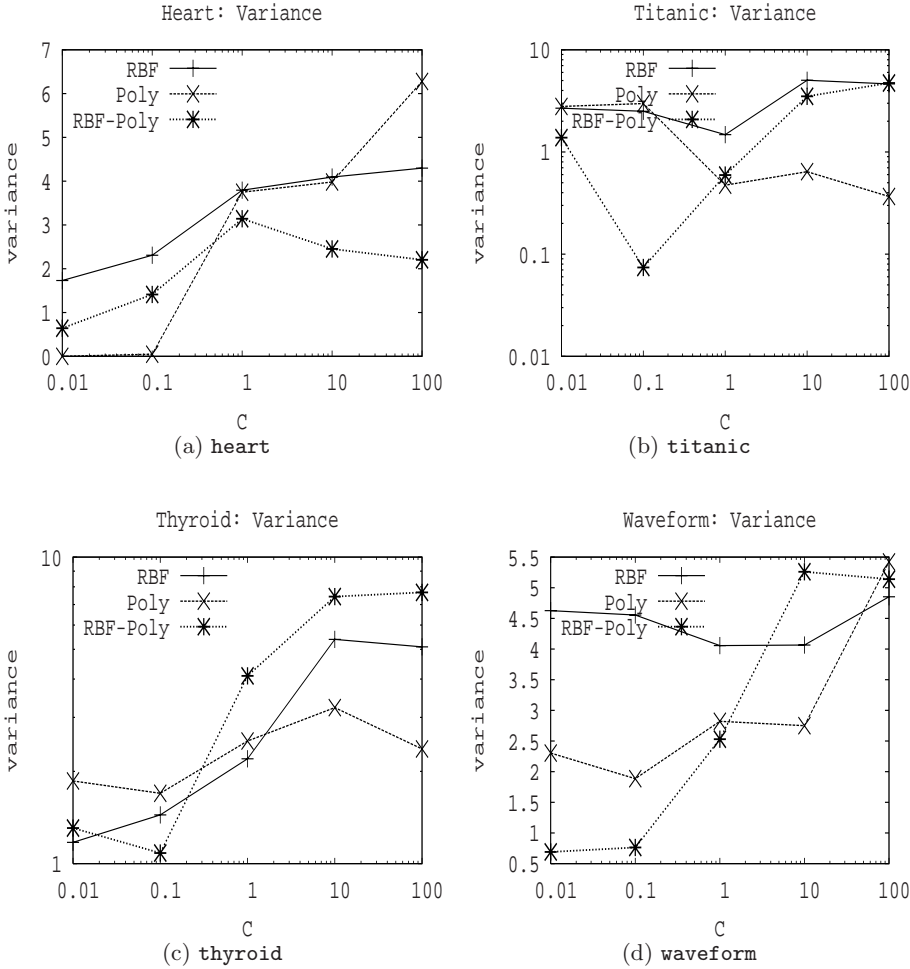
For different values of  $C$  and for all the three cases of the optimal kernel, stability analysis was carried out and the results are given in table 2. Each experiment is done as follows: for case (1) optimal kernel, for a particular value of  $C$  and for a data set, 100 LKMs are solved corresponding to the 100 bootstrap realizations and the following three quantities  $\widehat{err}(C)$ ,  $\widehat{err}(C_a)$  and  $\hat{\delta}$  are computed. Stability of LKMs is decided based on equation 5. Table 2 shows how many times LKM is found to be unstable for 5 different values of the user parameter  $C$ . Figures 1 (a) to (d) and 2 (e) shows the variation in error for different values of  $C$ . We note from these plots that the variance is high under small perturbations of the data set. For example, in the case of **heart** data set and with an optimal kernel of purely Gaussian kernel functions, LKM improved performance with bagging. This is given in table 3. From table 2, we observe that the stability factor is greater than 1 a majority of times.

Stability analysis in the case of SVMs has been carried out in [2]. It is shown experimentally that SVMs do not improve performance with bagging leading to stable algorithm behavior. Variance in the case of SVMs is reported close to 0 indicating that SVMs are *stable* algorithms.

### 6.2 Bias Variance Analysis

In bias-variance analysis, we study the following four points:

1. overfitting problem
2. comparison of bias, variance, un-biased variance, biased-variance, net-variance and number of support vectors between the three cases for the optimal kernel
3. average prediction error and
4. comparison of learning behavior between SVMs and LKMs.



**Fig. 1.** Variance behavior on heart, titanic, thyroid and waveform

1. **Overfitting:** In all the three cases for the optimal kernel and for different values of  $C$ , the bias, variance, unbiased-variance, biased-variance, net-variance and the number of support vectors were analyzed. The number of support vectors were observed to be high (figures 3 (d) and 4 (d)) and the averaged training error to be close to 0 and the bias is close to the error. From this we infer that LKMs **do not** overfit the given data set. In the literature it is argued that LKMs overfit the given data set [8]. In the case of SVMs, the experiments conducted by [10] show that overfitting occurs for small values of  $\sigma$  in the case of Gaussian kernels. For small values of  $C$ , LKMs have low bias and variance (as well as net variance) indicating that the learning of LKMs is not directly influenced by the user parameter. This is primarily due to the composition of the optimal kernel. This can be

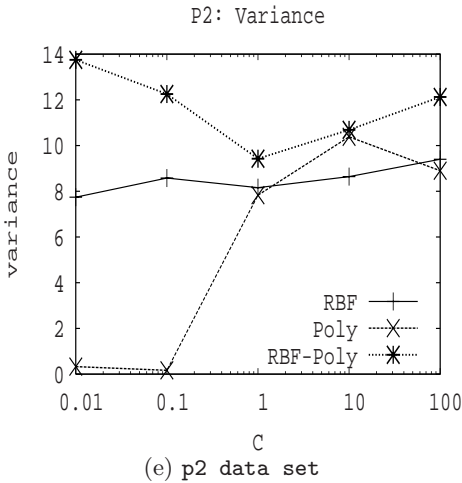
**Table 2.** Stability analysis results for different optimal kernel cases

Data set	Optimal kernel case		
	(1)	(2)	(3)
Heart	5/5*	2/5	3/5
Titanic	3/5	2/5	3/5
Thyroid	3/5	5/5	3/5
Waveform	2/5	5/5	4/5
p2	5/5	3/5	5/5

\*: Refer to Table 3 for details

**Table 3.** Stability analysis for **Heart** data set. Optimal kernel **case (i)**.

	C value				
	0.01	0.1	1.0	10.0	100.0
$\widehat{err}(C)$	0.123	0.124	0.138	0.149	0.153
$\widehat{err}(C_a)$	0.111	0.112	0.112	0.135	0.135
$\delta$	1.100	1.113	1.232	1.104	1.128
LKM stable?	unstable	unstable	unstable	unstable	unstable



**Fig. 2.** Variance behavior on **p2** data set

observed through the bias plots given in figures 3 (a) to (c) and 4 (a) to (c). In the case of SVMs, for small values of  $C$ , large bias is observed leading to the observation that SVMs do not learn for small values of  $C$ . The averaged bias values for all data sets closely follows the averaged error estimates of the respective data sets across all the three cases for the optimal kernel. At the same time, the averaged net-variance is close to 0, indicating that both unbiased-variance and biased-variance are close to each other. This is not

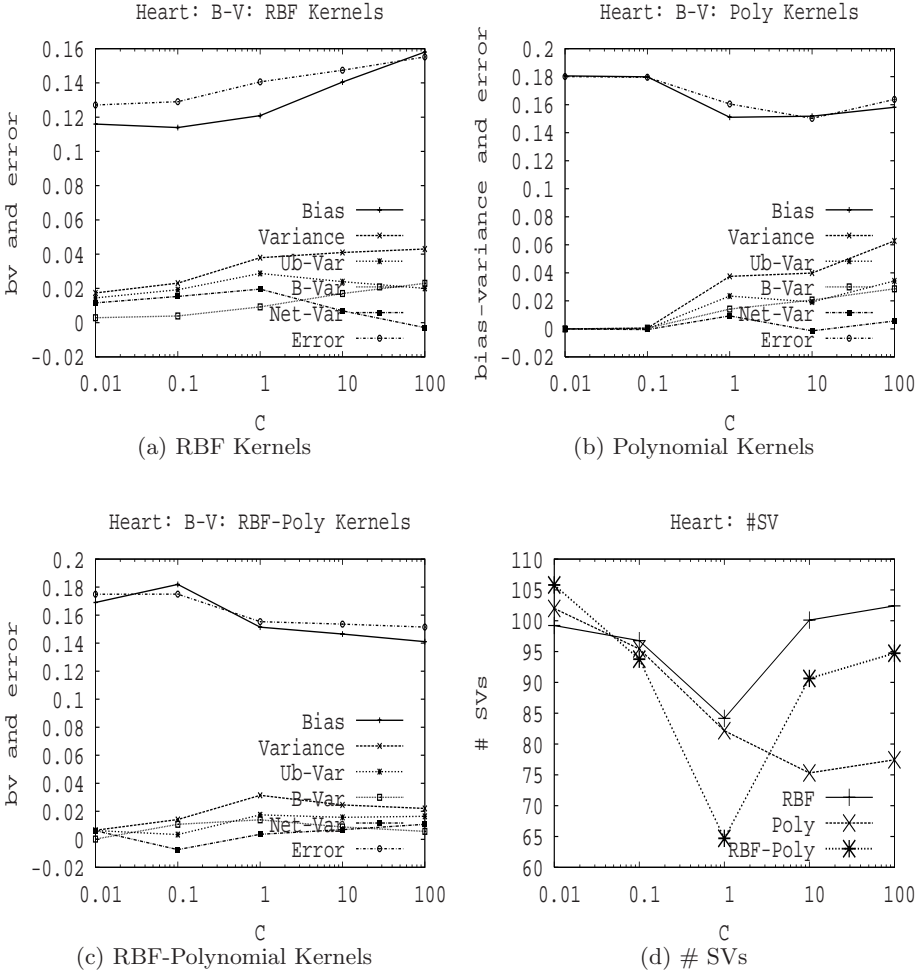
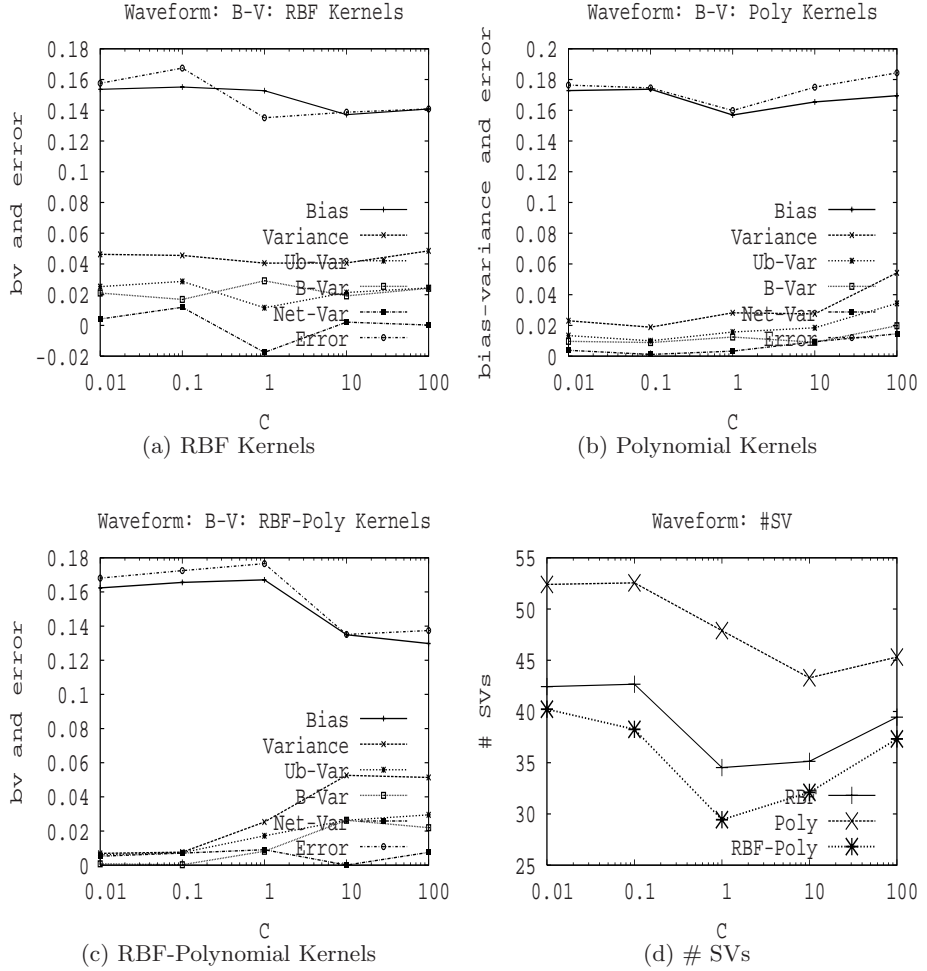


Fig. 3. Bias-Variance analysis for **Heart** data set

surprising as LKMs work in the transductive setting and the optimal kernel entries corresponding to the test data points are obtained for the *given* test data set.

2. **Comparing Three Cases of the Optimal Kernel:** Comparing the bias behavior across the different cases of the optimal kernel, in most instances a purely Gaussian kernel function combination has low bias compared to the other types of optimal kernel combinations. This is shown in figures 3 (a) to (c) and 4 (a) to (c). In the case of **heart** and **p2** data sets, there is a significant difference across different optimal kernel combinations. For rest of the data sets, no significant difference in bias is observed. We attribute the method of normalization in polynomial kernels as the primary reason.



**Fig. 4.** Bias-Variance analysis for Waveform data set

We observe large variations in biased-variance, unbiased-variance and net-variance for the above two data sets. On the number of support vectors front, purely polynomial kernel function case has more support vectors compared to other combinations of the optimal kernel and as shown in figures 3 (d) and 4 (d).

3. **Averaged Prediction Error:** The averaged prediction error for all the three cases of optimal kernel is shown in figures 3 (a) to (c) and 4 (a) to (c). From these graphs, we observe that the case (1) optimal kernel has smaller averaged prediction error compared to other cases in majority of experiments. This indicates that the Gaussian kernel combination is a better mixture for the data set at hand.

4. **SVMs Vs LKMs:** In the case of SVMs, bias behavior for the Gaussian kernels has three different regions (1) *high bias region*, (2) *transition region* and (3) *low bias region* as observed by [10]. However no such clear trend is observed in the case of LKMs. Similarly for polynomial kernels an ‘U’ shaped trend is observed for SVMs for the bias behavior [10] but no such trend is observed for LKMs.

## 7 Summary

In this work, we studied the stability and bias-variance of LKM algorithms through extensive experimentation carried out on 5 data sets with different types of optimal kernel combinations and compared the results for both stability and bias-variance. We observe that LKMs are unstable algorithms. They do not overfit the given data set as argued in the literature.

## References

1. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: bagging, boosting and variants. *Machine Learning*, 36:105–142, 1999.
2. I. Buciu, C. Kotropoulos, and I. Pitas. Demonstrating the stability of support vector machines for classification. *Signal Processing*, 86(9):2364 – 2380, 2006.
3. C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955 – 974, 1998.
4. Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1 - 3): 131 – 159, 2001.
5. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.
6. Sturm J. F. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11 - 12:625 – 653, 1999.
7. Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27 – 72, Jan 2004.
8. Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043 – 1071, Jul 2005.
9. G. Rätsch. Benchmark repository. Technical report, Intelligent Data Analysis Group, Fraunhofer-FIRST, 2005.
10. Giorgio Valentini and Thomas G. Dietterich. Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. *Journal of Machine Learning Research*, 5:725 – 775, 2004.
11. V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.
12. V. Vapnik and O. Chapelle. Bounds on error expectation for SVM. *Neural Computation*, 12:2013 – 2036, 2000.
13. Tong Zhang. Leave-one-out bounds for kernel methods. *Neural Computation*, 15:1397 – 1437, 2003.

# Query-Based Summarization of Customer Reviews

Olga Feiguina and Guy Lapalme

RALI

Département d'informatique et de recherche opérationnelle

Université de Montréal

CP 6128, Succ. Centre-Ville

Montréal, Québec, Canada, H3C 3J7

{feiguino,lapalme}@iro.umontreal.ca

**Abstract.** We describe an architecture for organizing and summarizing consumer reviews about products that have been posted on specialized web sites. The core technology is based on the automatic extraction of product features for which we report experiments on two types of corpora. We thus show that NLP techniques can be fruitfully used in this context for helping consumers sort out the mass of information displayed in such contexts.

## 1 Introduction

There are now many web sites that gather comments, written by customers, about certain products and services. These web sites are usually maintained either by manufacturers, by sellers (e.g. amazon.com) or by independent people (epinions.com) who (hope to?) make money by selling advertisement space that appears near the comments they display. These sites are quite useful for consumers because they provide *real users'* comments about products. These comments are often summarized by means of global scores or tables of features but the *real* information is still within texts for which there are yet few appropriate processing tools. The users' comments, being written by many different people, are often too numerous, repetitive and unclassified so finding information in this mass of diverse facts and anecdotes is quite an endeavor. As it often happens in our modern world, too much information is worse than not enough, and the time taken by many concerned users to write these comments is thus wasted because few people read and analyze them.

The goal of this paper is to show that it is possible to use NLP technology for organizing and summarizing these comments in order to better help potential buyers. We first describe the architecture of the system based on the automatic identification and merging of product features and then we present the experiments we have carried out and the results we have obtained. More detailed results are described in Feiguina [1].

## 2 Related Work

Customer reviews have received a lot of attention in the recent years. Research has mainly focused on extracting product features, identifying sentences commenting on them, and classifying reviews or sentences by attitude (positive/negative/neutral). Although our system builds on results obtained in sentiment classification research, we did not work on this aspect of processing customer reviews and focused instead on feature extraction and summarization.

Hu & Liu [2] proposed to summarize customer reviews for a given product by presenting a list of product features with the corresponding counts of positive/negative comments about them. When a user wants more detail, a long list of often repetitive sentences is returned and one of our goals was to solve this problem with our summarization approach. Hu & Liu [2] also worked on feature extraction using an association rule mining algorithm in conjunction with some frequency and overlapping based heuristics to extract the main product features. These features are then used to extract adjacent adjectives which are assumed to be opinion adjectives, which are in turn used to find features that were mentioned only once or several times. As Hu & Liu published their corpus, we could use it to develop some of our ideas.

Also using Hu & Liu's annotated corpus, Popescu and Etzioni [3] worked on feature extraction from reviews within the framework of *KnowItAll* [4]. They calculated point-wise mutual information (PMI) between all noun phrases found in customer reviews and phrases such as **scanner**, **scanner has**, etc. The PMI scores were then fed to a Naive Bayes classifier trained to decide if a given noun phrase is a product feature or not. PMI was calculated using the corpus of reviews as well as using the World Wide Web. The latter gave very good performance: 94% precision, 22% better than Hu and Liu [2], and 77% recall, 3% better than Hu and Liu [2].

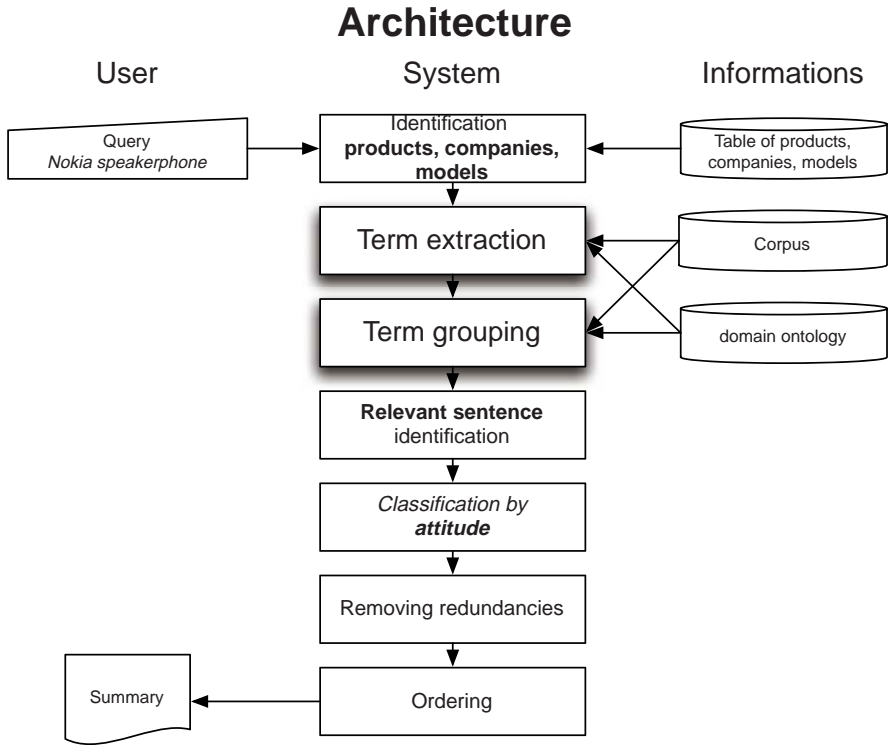
These works both relied on a syntactic analysis of the reviews; one of our goals was to limit ourselves to a shallow analysis in order to make the method more language independent. Hu and Liu used a small corpus of reviews while Popescu and Etzioni used the whole Web; our other goal was to strike a middle ground by using an unannotated corpus of reviews which is more *controlled* than the Web but while being able to gather more data than a manually annotated corpus.

Finally, Carenini [5] worked on mapping product features onto an existing taxonomy using various WordNet-based measures of semantic similarity [6]. We used his observations in our experiments on semantic grouping of automatically extracted features that will be presented in section 6.

## 3 Application Architecture

Figure 1 presents the architecture of our application that summarizes collections of customer reviews based on a query. We now give an overview of the modules





**Fig. 1.** Architecture of the customer comments summarizing application

with a simple example and show the output of our prototype at every stage. The application is invoked once a user enters a query, here *nokia speakerphone*.

**Identification of product, company, model.** Identify, based on the query, what product the user is talking about. In order to do this, we compiled a small database that links company names to products and models of products. Entries of the database are triples of the form {*company, product, model*} (e.g. {*Canon, digital camera, Powershot SD300*}) for which we allow slight variations such as *Cannon* for *Canon*, etc. Having a pre-compiled list of associated companies, products and models allows for some consistency checks that ensure the user is not mixed up about what a certain company manufactures, what products exist or what product comes in what models. For our example, we extract *nokia* as the company name and no product or model. What remains of the query is *speakerphone*.

**Identification of product features.** Having identified as much as possible the product, company and model information, we use our lists of features

(their creation will be described in Section 5) to identify the ones the user is interested in. For example, given the query **nokia cellphone charger**, we first identify **nokia** as the company, **cellphone** as the product, and then use the list of cellphone features to identify the feature **charger**. In our example, we extract **speakerphone**.

**Extraction of relevant sentences.** We assume that customer reviews are already labeled with the company, product and model they describe. Using the information extracted from the query, we identify the relevant reviews. From them, we extract not only sentences containing features as stated in the query, but also their synonyms. For example, in the electronics domain, for the query **screen**, sentences mentioning **display** are also relevant.

**Classification of sentences by attitude.** The next step is to classify relevant sentences as being either positive or negative. In our prototype, we used a very simple minded approach to this problem: we only label a sentence as positive or negative if the annotation (to be presented in section 4) of a sentence labels it as entirely positive or negative.

**Elimination of redundant sentences.** We define two sentences as redundant if they describe the same features with the same attitude. For example, **the speakerphone works better than any speakerphone i've ever had .** and **this phone has a very cool and useful feature - the speakerphone .** are both positive and talk about the **speakerphone**, so we would consider them redundant. Of course some information is lost through the elimination of sentences, but the essence of the feature being commented on positively is preserved and expressed in natural language.

Often, several features are commented upon in the same sentence. Although the user may not be interested in features not mentioned in the query, we keep the other features. In order to judge if the relevant sentences obtained thus far contain redundant sets, we first extract features other than those mentioned in the query from each sentence. This is done using our pre-compiled list of features via simple pattern matching. Given two redundant sentences, we keep the longer one, unless one of them has already been judged redundant with another sentence and kept, in which case we keep it again.

**Re-ordering of the remaining sentences.** The remaining sentences form the summary. We group sentences by their attitudes (positive or negative) and we first present sentences that cover more aspects, i.e. more sentences are redundant with those. The result for our example is presented in Figure 3.

Within this overall architecture of a system that summarizes customer reviews based on a query, we mainly focused on the automatic feature extraction that we describe in sections 5 and 6. The next section presents the corpora used in the experiments.

Speakerphone - loud and clear has some nice extra features like currency converter and a stopwatch. My favorite features, although there are many, are the speaker phone, the radio and the infrared. The speaker phone is very functional and I use the phone in the car, very audible even with freeway noise. [...] The phone book is very user-friendly and the speakerphone is excellent. The phone has great battery life, fm radio, excellent signal, hands free speakerphone (which I have to say is probably my favorite function) and downloadable java apps. [...] Only one complaint about the speakerphone, you can only activate the speakerphone feature once the person you are calling answers the phone, not while the phone is ringing. Great speakerphone and great reception !! recommend! The speakerphone: People I talk to on the speakerphone are shocked when the phone comes out at times that I'm even using a speakerphone.

**Fig. 2.** Excerpts from a summary produced from the query `Nokia speakerphone`

## 4 Corpora and Annotation

In our experiments, we used two corpora, see Table 1.

**HL corpus** created and annotated by Hu&Liu [2] described in Section 2.

**SK corpus** was compiled for us by Shahzad Khan (PhD candidate at Cambridge University) who wrote a crawler for [www.epinions.com](http://www.epinions.com). Those reviews were manually sorted by product.

We first relied on the annotation of the HL corpus and we will present results of our experiments using it in the next section. Unfortunately, we noted a series

**Table 1.** The HL and SK corpora: statistics on their size and the description of the annotations in the HL corpus

Product	HL corpus		SK	
	# reviews	≈ # words	# reviews	≈ # words
Digital camera	79	235	10605	340
Cellphone	41	235	4511	460
MP3 player	95	340	4042	370
DVD player	99	130	3266	280
	314	235	22424	360

### Annotation Description

[t]	review titles
xxxx[+ -n]	xxxx is a product feature + - : positive or negative opinion n strength of the opinion
xxxx[u]	the product feature xxxx is not explicitly mentioned
xxxx[p]	anaphora resolution is necessary to determine the feature

of issues in their annotation that interfered with our approach. Some of them are illustrated by the following examples, all from the first 72 lines of the MP3 player section of the corpus:

- Although implicit features are supposed to be labeled with [u], this is not always the case: `size[-1]##it could be a little bit bigger , but it 's easy to get used to .`
- Sometimes non-features (looking) are extracted: `looking[+2]##by the way , it looks nice also .` In this example, looking is not in the sentence but is not marked [u].
- Features that do not exist are not extracted: `##( i would have appreciated having a firewire plug , however )`. Although future and desired features have a different status, if people comment on them, they need to be on the list of features the system knows about.
- Features are sometimes omitted, so if our extractor found them, they would be counted as errors if we were using this annotation to evaluate it: `##i probably would have liked to have a player in something other than silver / metallic ... like the battery adapters on their usb thumbdrive ( muvo nx ) MP3 player models . (battery adapters not extracted)`  
`##since the front plate is removable to access the battery compartment , aftermarket alternate covers would not be difficult or expensive to make . (front plate not extracted)`

Because of these problems, we also compiled our own list of product features. We extracted them from the first 777 sentences of the MP3 section of the HL corpus. Our total was 221 features that we will call the FL (Feiguina and Lapalme) set, as opposed to the HL set extracted from Hu and Liu's annotation. To give a numerical comparison, in the HL set, the total number of features extracted from the whole MP3 section of the corpus (1811 sentences) was only 181.

## 5 Feature Extraction

The goal of our feature extraction experiments was to extract a set of product features, as complete and noise-free as possible, from a set of customer reviews describing a certain product (cellphones, MP3 players, etc).

### 5.1 Method

Our method is based on the observation that patterns can be found in the way customers comment on product features. One common pattern is the *feature is adjective* (e.g. the speakerphone is great). This approach is intuitive, requires only a shallow parsing and can make use of large unannotated corpora.

In order to identify useful patterns automatically, we used our annotation of the MP3 section of the HL corpus presented in section 4 and a terminology extractor.

Our terminology extractor, written by Patry and Langlais [7], takes a training text, a set of term examples extracted from this text by a human and a corpus to extract new terms from. It then executes the following steps:

1. Perform part-of-speech tagging of the training text, the training terms and the corpus.
2. Convert the training text to a stream of part-of-speech tags.
3. Learn a language model of terms which represents the most likely part-of-speech sequences within terms.
4. Use the language model for extracting new term candidates from an unseen corpus.
5. Score the terms using AdaBoost on other features such as length, frequency in the corpus, etc.

We used this terminology extractor with the MP3 section of the HL corpus as training text. We used both the features we extracted manually and the HL annotation as training terms. In addition to the term, we used the words around the extracted features.

For example, if from **The silver screen is one of the best screens I've ever had.** we extracted the feature **screen**, we would feed to the terminology extractor **silver screen** or **screen is one** or **silver screen is** as an example of a term depending on the type of context we're considering. The corpus to extract from was the unannotated SK corpus for a given product.

The extracted terms were then *cleaned* from the context. For example, if the context was **WORD TERM WORD (silver screen is)**, and **clear sound is** were extracted, we would then remove the context to get **sound. Olga:vérifier le premier exemple...**

## 5.2 Results

In our experiments, we were limited by the need to evaluate manually the terms extracted, so we worked with just three types of context: **TERM WORD WORD** (**screen is one**), **WORD TERM WORD WORD** (**silver screen is one**), and **WORD WORD TERM** (**the silver screen**) and just one product: (Nokia) cellphones.

The extracted terms were evaluated by the first author only, so the scores given in table 2 are approximations of the true precision. For some large sets of terms, we did not evaluate the whole set if the precision stayed the same or got worse as evaluation went on. Our examination of scores produced by the terminology extractor showed that they do not help eliminate the non-features from the list of results.

Our experiments show the potential value of context for feature identification. We have shown that more elaborate contexts (*word term word word* give good accuracy but very low recall; perhaps multiple elaborate contexts can be combined to give a better performance. Of the less restrictive contexts, *term word word* performed the best; patterns like *term is adj* were very fruitful, like we expected.

**Table 2.** Terminology extraction experiments

Training corpus	context	# extracted terms	estimated precision
HL		12 000	30%
HL	<i>word term word</i>	430	55%
HL	<i>term word word</i>	810	65%
HL	<i>word term word word</i>	53	85%
FL	<i>word term word</i>	1200	10%
<b>FL</b>	<b><i>term word word</i></b>	<b>800</b>	<b>80%</b>
FL	<i>word term word word</i>	26	88%

### 5.3 Future Work

Our experiments show that the approach is promising, but much remains to be explored. In particular, it would be interesting to try different types of context and to study the generality of acquired patterns by verifying if patterns learned from a MP3 player customer reviews can be useful for non-electronics products. We should also use stricter annotation and evaluation procedures with multiple people so that inter-annotator and evaluator agreement can be measured.

## 6 Semantic Grouping of Features

Having extracted product features, it is desirable to group them by semantic similarity. For example, features related to the **screen** should be grouped with the ones related to the **display**. Moreover, it would be interesting to identify more distant similarity such as **email** and **messaging**. Feature grouping is especially important given our approach to feature extraction, where features like **color screen** and **outer screen** are extracted: the price for extracting complete features is that there is a multitude of them and many of them are similar.

### 6.1 Grouping by Salient Words

We carried out two types of preliminary experiments. A first approach was to group extracted features by words they contain, considering only the words salient for the domain. We used TF-IDF ( $\text{Score}(\text{word}) = \text{frequency in SK corpus} / \text{frequency in Hansard}$ ) to determine the salient words, those whose score was above an empirically set threshold. Of the 242 words thus found, 100 were in at least one extracted feature; we will call them 'keywords'. Using keywords to group the features resulted in groups presented in Table 3.

These groups show that our method allows to group related features such as **color screen** and **outer screen**. Our next goal is to establish links between groups based on synonymous words, such as **screen** and **display**.

Table 3. Some groupings of features

email	email, email feature, short emails, email client, writing emails
battery	battery door, battery time, battery power, battery capacity, life battery, battery consumption, battery cover, battery use, battery indicator, rechargeable battery, lithium battery, battery line, battery meter, actual battery, polymer battery, big battery, battery life, battery quality, battery compartment, standard battery, battery level
id	id display, caller id, picture id, id screen, id, id feature
favorite	personal favorite
microphone	external microphone, microphone combo, microphone gain, microphone quality
ericsson	ericsson
charger	extra charger, desktop charger, charger connector, car charger, top charger, charger device, base charger, charger input, travel charger

6.2 Grouping by Semantic Similarity

To this end, we attempted to use WordNet-based measures of semantic similarity. We used *path*, *lin*, and *res* similarity measures [6].

As usual when dealing with lexical resources, the first problem to solve was word sense disambiguation. We implemented a simple algorithm for choosing a word sense based on its similarity with a set of domain keywords. From the list of 100 words with high TF·IDF scores that occur in at least one extracted feature, 12 had only one WordNet sense. They were: {keypad, speakerphone, pda (personal digital assistant), phonebook, headset, handset, faceplate, voicemail, email, earpiece, messaging, modem}.

The quality of this WSD algorithm, evaluated using 125 words disambiguated manually, is shown in Table 4.

Table 4. The performance of our word sense disambiguation (WSD) algorithm using three different measures of semantic similarity, evaluated on 125 words with high TF·IDF scores

Sim measure	WSD precision
path	51%
res	47%
lin	55%

Using this WSD module, we used the following algorithm after having computed the semantic similarity of all feature pairs (if a feature consists of multiple words, an average over the semantic similarity of all pairs of words is taken).

- If two keywords have semantic similarity greater than an empirically determined threshold  $T$ , we merge the feature groups of these keywords.
- If a non-grouped feature has semantic similarity greater than  $T$  with a keyword, add it to that keyword's group.
- If a feature has semantic similarity greater than  $T$  with another grouped feature, add it to all of that feature's groups.
- If two non-grouped features have semantic similarity greater than  $T$ , create a new group with those features in it.

We iterate over these steps until no further changes are made to maximize the grouping. The resulting grouping steps were evaluated manually by the first author. The best performance of about 67% was observed when the measure of similarity *path* was used; of the 55 proposed groupings, 27 seemed good to us. Some examples of the groupings we judged good/bad are as follows:

- Good groupings:
  - **switch**, **button** (*two keywords*)
  - **interior panel**, **inside panel** (*two ungrouped features*)
  - **net access**, **internet** (*an ungrouped feature and a keyword*)
  - **message system**, **voice message** (*an ungrouped feature and a grouped feature*)
- Bad groupings:
  - **device**, **alarm** (*two keywords*)
  - **list**, **menu** (*an ungrouped feature and a keyword*)
  - **input system**, **message system** (*an ungrouped feature and a grouped feature*)

Although the good groupings look promising, there is a number of necessary improvements to eliminate the bad ones. Many bad groupings are the result of hyponyms and hypernyms getting high semantic similarity scores (e.g. **device** and **alarm**). As neither **path** nor **res/lin** take into account this issue, they all do poorly at grouping features. The fact that **path** is somewhat better does not seem related to its ability to take into account hypernym/hyponym relationships. These bad groupings could be handled by creating a special measure of semantic similarity where such relationships between words receive a lower score. Alternatively, a clustering algorithm more sophisticated than our threshold-based one should be investigated. Although the groupings generated by our algorithm are not enough for automatic ontology construction, they offer a great start to a human ontology engineer.

## 7 Future Work

An interesting extension to add to our architecture would be a *cooperative analysis* such as the one proposed by Benamara and St-Dizier [8] to ensure that the initial query is coherent, i.e. that the product or the features really relate to the right manufacturer. This would imply some consistency checks with a



database of products with their set of features and their manufacturer. This type of database is already present for other aspects of these types of customer sites.

We have done experiments only with electronics related products for which our patterns seemed to apply, but it would be interesting to apply them to comments on other types of products or on bigger corpora to see how such an approach would scale up. It would be even more challenging to test these summaries with humans or compare them with the current types of summaries found on consumer web sites.

## 8 Conclusion

We have shown in this paper how to organize consumers' comments based on an automatic term extraction mechanism and a grouping around issues that were deemed relevant in these comments. Of course, it remains to be seen if such a type of organization is really useful for consumers. To our knowledge, this work is the first attempt to produce natural language summaries for this type of texts. We have shown that although these free comments are written by non specialists, they seem to use similar text pattern that allowed us to identify the mains points described by these texts, to regroup similar texts while removing the ones that merely repeat already reported information.

## Acknowledgment

We would like to thank Shahzad Khan for allowing us to use his web crawler. We thank also Balázs Kégl for discussions and comments about machine learning algorithms. We also thank the members of the RALI laboratory in particular Philippe Langlais and Alexandre Patry who developed the term extractor we used in our experiments and Fabrizio Gotti for his help on many technical and scientific matters. This work has been partly funded by a NSERC discovery grant.

## References

1. Feiguina, O.: Automatic summarization of customer reviews. Master's thesis, University of Montreal (2006)
2. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 2004), New York, NY, USA, ACM Press (2004) 168–177
3. Popescu, A.M., Etzioni, O.: Extracting product features and opinions from reviews. In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Vancouver, British Columbia, Canada, Association for Computational Linguistics (October 2005) 339–346
4. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence* **165**(1) (2005) 91–134

5. Carenini, G., Ng, R.T., Zwart, E.: Extracting knowledge from evaluative text. In: Proceedings of the 3rd international conference on Knowledge capture (K-CAP 2005), New York, NY, USA, ACM Press (2005) 11–18
6. Patwardhan, S., Banerjee, S., Pedersen, T.: Using measures of semantic relatedness for word sense disambiguation. In: Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City, Mexico (2003) 241–257
7. Patry, A., Langlais, P.: Corpus-based terminology extraction. In: Proceedings of the 7th International Conference on Terminology and Knowledge Engineering, Copenhagen, Denmark (August 2005) 313–321
8. Benamara, F., Saint-Dizier, P.: Construction de réponses coopératives : du corpus à la modélisation informatique. In: *Revue Québécoise de linguistique*. Number 3 in 18. (septembre 2004) 34–59

# Multi-state Directed Acyclic Graphs

Michael Wachter<sup>1</sup> and Rolf Haenni<sup>1,2</sup>

<sup>1</sup> University of Bern, Switzerland  
`{wachter,haenni}@iam.unibe.ch`

<sup>2</sup> Bern University of Applied Sciences, Switzerland  
`rolf.haenni@bfh.ch`

**Abstract.** This paper continues the line of research on the representation and compilation of propositional knowledge bases with propositional directed acyclic graphs (PDAG), negation normal forms (NNF), and binary decision diagrams (BDD). The idea is to permit variables with more than two states and to explicitly represent them in their most natural way. The resulting representation languages are analyzed according to their succinctness, supported queries, and supported transformations. The paper shows that most results from PDAGs, NNFs, and BDDs can be generalized to their corresponding multi-state extension. This implies that the entire knowledge compilation map is extensible from propositional to multi-state variables.

## 1 Introduction

Boolean functions play a crucial role in many areas of computer science and mathematics, most notably in Artificial Intelligence, digital system design, formal verification, mathematical logic, reliability theory, and combinatorial optimization. They are fundamental whenever knowledge is represented by propositional variables, i.e. through a set of possible states in the corresponding multi-dimensional Boolean space.

In practice, working with Boolean functions presupposes efficient ways to represent them. Among the existing approaches for representing Boolean functions are truth tables, Karnaugh maps, sum-of-products such as DNFs or prime implicants, product-of-sums such as CNFs or prime implicates, and most notably *binary decision diagrams* (BDD) [1,2,3], *negation normal forms* (NNF) [4,5], *propositional directed acyclic graphs* (PDAG) [6], and all their derivatives. Some of these forms are known to be impractical, as they impose representations of exponential size for *most* possible  $r$ -ary functions [7], but many BDD, NNF, and PDAG forms provide polynomial representations at least for *many* functions.

The restriction of these techniques to propositional variables does not entirely meet the requirements of real-world models, which are often not limited to Boolean variables. For example, the possible states of a traffic light (in most parts of the world) are red, yellow, and green. At a particular time, the traffic light is in exactly one of these states. We will use the following terminology to distinguish the different types of variables: *propositional*, *Boolean*, or *binary* variables have exactly two states, *non-binary* variables have more than two states,

and *multi-state* variables have two or more states.<sup>1</sup> In addition, we suppose that each (binary, non-binary, or multi-state) variable has a unique (but typically unknown) *true* state.

Multi-state variables have been discussed in the literature of *decision diagrams* [10], where *multivalued decision diagrams* (MDD) arise as an extension of BDDs to multi-state variables. They are an alternative to the usual replacement of multi-state variables by  $\lceil \log_2 \ell \rceil$  Boolean variables, where  $\ell$  denotes the number of possible states. In this way, MDDs can be transformed into BDDs with a linear growth in size, which relativizes the benefits of MDDs over BDDs, especially if  $\ell$  is small. The conclusion in [10] is the following:

“MDDs are useful if the considered function has a natural description with multivalued variables.” [10, Section 9.1, page 216]

In applications of NNFs, especially in the contexts of probabilistic reasoning, Bayesian networks, and model counting [11,12,13], it is common to use similar Boolean encodings for multi-state variables. These encodings typically use  $\ell$  (or  $\ell - 1$ ) auxiliary Boolean variables, i.e. one for each state (except for the last one). The exclusivity and exhaustiveness of these auxiliary variables requires explicit representations of corresponding *exclusive ORs*, which is a non-negligible overhead, especially if  $\ell$  is large. Another problem of these Boolean encodings is the computation of probabilities, if independent probability mass functions are given for all multi-state variables. The core of the problem is the fact, that the auxiliary Boolean variables are no longer independent. It is possible to overcome this difficulty by transforming the given probabilities into conditional probabilities [11,12], but the existing solutions are rather cumbersome.

If we decide to work with multi-state variables from the beginning, these problems all disappear, including the one of selecting an appropriate Boolean encoding. The goal of this paper is thus to modify the existing PDAG, NNF, and BDD languages to multi-state variables (unless it is not yet done elsewhere). We will show that most theoretical results remain valid. In this way, we add an additional dimension to the knowledge compilation map promoted in [5,6].

The remainder of this paper is organized as follows. In Sect. 2, we extend the definition of PDAGs to multi-state variables, compare the resulting *multi-state directed acyclic graphs* (MDAG) with PDAGs, and finally define different MDAG sub-languages. In Sect. 3, some theoretical results about the succinctness, the supported queries, and the supported transformations of PDAGs are generalized to MDAGs. Section 4 concludes the paper.

## 2 Multi-state Directed Acyclic Graph

Let  $\mathbf{V} = \{V_1, \dots, V_r\}$  be a set of  $r$  variables and suppose that  $\Omega_{V_i}$  denotes the finite set of states of  $V_i$ . A finite indicator function  $f$  is defined by  $f : \Omega_{\mathbf{V}} \rightarrow \mathbb{B}$ ,

<sup>1</sup> To outline the difference to *multivalued* or *many-valued* logics, where logical sentences are mapped into more than two truth values [8,9], we prefer to use the term ‘multi-state’ instead of ‘multivalued’ or ‘many-valued’.

where  $\Omega_{\mathbf{V}} = \Omega_{V_1} \times \cdots \times \Omega_{V_r}$  and  $\mathbb{B} = \{0, 1\}$ . To emphasize the fact that  $f$  is a mapping from the Cartesian product  $\Omega_{V_1} \times \cdots \times \Omega_{V_r}$  to  $\{0, 1\}$ , we will call it a *Cartesian indicator function* (CIF). The so-called *satisfying set*  $S_f = \{\mathbf{x} \in \Omega_{\mathbf{V}} : f(\mathbf{x}) = 1\} = f^{-1}(1)$  of  $f$  is the set of  $r$ -dimensional vectors  $\mathbf{x} \in \Omega_{\mathbf{V}}$  for which  $f$  evaluates to 1. Special cases of finite CIFs are Boolean functions (BF), where  $\Omega_{V_i} = \mathbb{B}$ , and therefore  $\Omega_{\mathbf{V}} = \mathbb{B}^r$ .

The most general forms for representing BFs are PDAGs. As shown in [6], the well-known NNFs, BDDs, and their derivatives correspond to subsets of PDAGs. While *multivalued decision diagrams* (MDD) have been proposed as an extension of BDDs to multi-state variables in the context of *decision diagrams* [10], there is no such extension for NNFs or PDAGs. PDAGs, and therewith NNFs, will be extended to multi-state variables below. We will see that the resulting language also includes the existing MDDs. As in the case of PDAGs and NNFs, the representation we propose here is based on directed acyclic graphs, but now we impose some particularities.

**Definition 1.** A multi-state DAG (MDAG) is a rooted directed acyclic graph, where:

1. Leaves are represented by  $\square$  and labeled with  $\top$  (true),  $\perp$  (false), or  $X=x$ , where  $X \in \mathbf{V}$  is a variable and  $x \in \Omega_X$  is one of its states;
2. Non-leaves are represented by  $\Delta$  (logical and),  $\nabla$  (logical or), or  $\Diamond$  (logical not).  $\Delta$ - and  $\nabla$ -nodes have at least one child,  $\Diamond$ -nodes have exactly one child.

In a MDAG, each node  $\alpha$  represents a finite CIF  $f_\alpha$  by

$$f_\alpha = \begin{cases} \bigwedge_{i=1}^t f_{\beta_i} = \min_{i=1}^t f_{\beta_i}, & \text{if } \alpha \text{ is an } \Delta\text{-node with children } \beta_1, \dots, \beta_t, \\ \bigvee_{i=1}^t f_{\beta_i} = \max_{i=1}^t f_{\beta_i}, & \text{if } \alpha \text{ is an } \nabla\text{-node with children } \beta_1, \dots, \beta_t, \\ \neg f_\psi = 1 - f_\psi, & \text{if } \alpha \text{ is a } \Diamond\text{-node with the child } \psi, \\ 1, & \text{if } \alpha \text{ is a } \square\text{-node labeled with } \top, \\ 0, & \text{if } \alpha \text{ is a } \square\text{-node labeled with } \perp, \\ f_{X=x}, & \text{if } \alpha \text{ is a } \square\text{-node labeled with } X=x, \end{cases}$$

where  $f_{X=x}(\mathbf{x})$  with  $\mathbf{x} \in \Omega_{\mathbf{V}}$  is defined by

$$f_{X=x}(\mathbf{x}) = \begin{cases} 1, & \text{if } x \text{ is the corresponding value of } X \text{ in } \mathbf{x}, \\ 0, & \text{otherwise.} \end{cases}$$

The MDAG depicted in Fig. 1 represents the finite CIF  $f = ([Y=y_1] \wedge [X=x_1]) \vee ([Y=y_2] \wedge \neg[X=x_2]) \vee ([X=x_2] \wedge [Y=y_3])$ . Note that with this,  $\Omega_X$  and  $\Omega_Y$  are not necessarily restricted to  $\{x_1, x_2\}$  and  $\{y_1, y_2, y_3\}$  from the beginning.

Formally, we will write  $\mathbf{MDAG}_{\mathbf{V}}$  for the set of all possible MDAGs with respect to  $\mathbf{V}$ . We follow the view from [5,6] and call  $\mathbf{MDAG}_{\mathbf{V}}$  a *language*. When no confusion is anticipated, we omit the reference to the set  $\mathbf{V}$ , i.e. we simply write MDAG

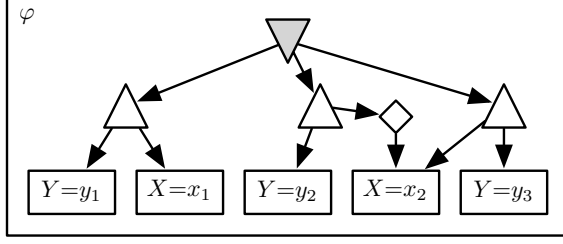


Fig. 1. The finite CIF  $f$  represented as the MDAG  $\varphi$

instead of  $\text{MDAG}_{\mathbf{V}}$  and  $\Omega$  instead of  $\Omega_{\mathbf{V}}$ . Our convention is to denote MDAGs by lower-case Greek letters such as  $\varphi$ ,  $\psi$ , or the like. Remember that any node  $\alpha$  included in a MDAG  $\varphi$  defines its own (sub-) MDAG, and is thus another element of  $\text{MDAG}$ .

The number of edges of  $\varphi \in \text{MDAG}$  is called its *size* and is denoted by  $|\varphi|$ . MDAGs are called *binary*, if no  $\Delta$ - or  $\nabla$ -node has more than two children. The set of variables included in a sub-MDAG  $\alpha$  of  $\varphi$  is denoted by  $\text{vars}(\alpha)$ . The *path-length* of a path from the root to a leaf is the number of edges minus the number of  $\Diamond$ -nodes along the path. The *height* of  $\varphi$ , denoted by  $h(\varphi)$ , is its maximal path-length. Note that these concepts (size, binary, *vars*, path-length, height) have the same meaning for PDAGs.

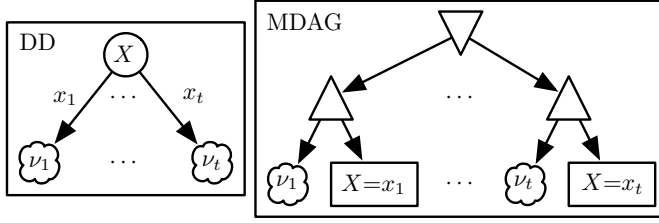
Any finite CIF can be represented by a MDAG, so the  $\text{MDAG}$  language is *complete*. On the other hand, MDAGs are not *canonical*, i.e. we may have several equivalent MDAGs representing the same finite CIF. Two MDAGs  $\varphi, \psi \in \text{MDAG}$  are *equivalent*, denoted by  $\varphi \equiv \psi$ , iff  $f_{\varphi}(\mathbf{x}) = f_{\psi}(\mathbf{x})$  for all  $\mathbf{x} \in \Omega$ . Furthermore,  $\varphi$  *entails*  $\psi$ , denoted by  $\varphi \models \psi$ , iff  $f_{\varphi}(\mathbf{x}) \leq f_{\psi}(\mathbf{x})$  for all  $\mathbf{x} \in \Omega$ . In terms of their satisfying sets,  $S_{f_{\varphi}} = S_{f_{\psi}}$  means equivalence and  $S_{f_{\varphi}} \subseteq S_{f_{\psi}}$  entailment. Again, both equivalence and entailment have the same meaning for PDAGs.

## 2.1 Sub-languages

We will now turn our attention to some sub-languages of  $\text{MDAG}$ . The classification of sub-languages is done according to the following properties, which are based on the ones given in [5,6]:

1. *Decomposability*: the sets of variables of the children of each  $\Delta$ -node  $\alpha$  in  $\varphi$  are pairwise disjoint (i.e. if  $\beta_1, \dots, \beta_n$  are the children of  $\alpha$ , then  $\text{vars}(\beta_i) \cap \text{vars}(\beta_j) = \emptyset$  for all  $i \neq j$ );
2. *Determinism*: the children of each  $\nabla$ -node  $\alpha$  in  $\varphi$  are pairwise logically contradictory (i.e. if  $\beta_1, \dots, \beta_n$  are the children of  $\alpha$ , then  $S_{f_{\beta_i}} \cap S_{f_{\beta_j}} = \emptyset$  for all  $i \neq j$ );
3. *No-Negation*:<sup>2</sup>  $\varphi$  does not contain any  $\Diamond$ -node;

<sup>2</sup> No-negation corresponds to *simple-negation* in [5,6].



**Fig. 2.** A decision node  $X$  with  $\Omega_X = \{x_1, \dots, x_t\}$  and its decision structure.  $\nu_i$  are further nodes of the DD, resp. MDAG.

4. *Flatness*:  $h(\varphi) \leq 2$ ;
5. *Simple-Conjunction*: the children of each  $\Delta$ -node  $\alpha$  in  $\varphi$  are leaves without any common variable (i.e.  $\alpha$  is a proper *term*),
6. *Simple-Disjunction*: the children of each  $\nabla$ -node  $\alpha$  in  $\varphi$  are leaves without any common variable (i.e.  $\alpha$  is a proper *clause*);
7. *Smoothness*: the children of each  $\nabla$ -node  $\alpha$  in  $\varphi$  include the same set of variables (i.e. if  $\beta_1, \dots, \beta_n$  are the children of  $\alpha$ , then  $\text{vars}(\beta_i) = \text{vars}(\beta_j)$  for all  $i \neq j$ ).

Note that smoothness is not that important from a complexity viewpoint, unless we have flatness [5]. We will not further discuss smoothness in this paper. Simple-disjunction and simple-conjunction are characteristic for classical forms such as CNFs, DNFs, prime implicates, etc.

In addition to the basic properties above, we also consider some properties of decision diagrams (DD). According to [10,14], a decision diagram consists of non-leaves, so-called *decision nodes*, and leaves, so-called *terminals*. Decision nodes are represented by  $\bigcirc$ , labeled with  $X \in \mathbf{V}$ , and have outdegree  $|\Omega_X|$ , see left part of Fig. 2. In addition, Fig. 2 shows the one-to-one mapping between a decision node and its MDAG representation, called *decision structure*. Terminals are represented by  $\square$  and labeled with 1 or 0 in decision diagrams, resp. with  $\top$  or  $\perp$  in MDAGs.

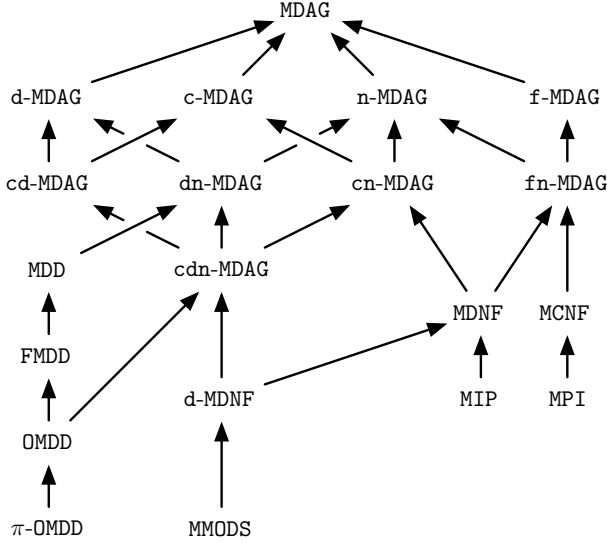
8. *Decision*:  $\varphi$  contains only decision structures and terminals.
9. *Read-once*: each path from the root to a terminal contains at most one decision node/structure for each variable  $X \in \mathbf{V}$ ;
10. *Ordering*: on each path from the root to a leaf, the occurrence of decision structures respects a total ordering on  $\mathbf{V}$ ;
11.  *$\pi$ -Ordering*: on each path from the root to a leaf, the occurrence of decision structures respects the globally specified ordering  $\pi$  on  $\mathbf{V}$ .

The sub-languages of MDAG are defined via these properties in the same way as sub-languages are defined for PDAG [6]. Table 1 shows MDAG and some of its most

**Table 1.** MDAG and some of its sub-languages according to the 11 properties. With ‘x’ we indicate that a language satisfies the corresponding property, and ‘(x)’ means that this property is implied by other properties.

	decomposability	determinism	no-negation	flatness	simple-conjunction	simple-disjunction	smoothness	decision	read-once	ordering	$\pi$ -ordering	Description	Boolean Case
MDAG												multi-state directed acyclic graph (MDAG)	PDAG
c-MDAG	x											decomposable MDAG	c-PDAG
d-MDAG		x										deterministic MDAG	d-PDAG
n-MDAG			x									negation-free MDAG	NNF ( $\equiv$ n-PDAG)
cd-MDAG	x	x										decomposable deterministic MDAG	cd-PDAG
cn-MDAG	x		x									decomposable negation-free MDAG	DNNF ( $\equiv$ cn-PDAG)
dn-MDAG		x	x									deterministic negation-free MDAG	d-NNF ( $\equiv$ dn-PDAG)
cdn-MDAG	x	x	x									decomposable deterministic negation-free MDAG	d-DNNF ( $\equiv$ cdn-PDAG)
f-MDAG			x	x								flat MDAG	f-PDAG
fn-MDAG		x	x	x								flat negation-free MDAG	f-NNF ( $\equiv$ fn-PDAG)
MCNF		x	x	x	x							multi-state conjunctive normal form (MCNF)	CNF
MPI		x	x	x	x							multi-state prime implicates	PI
MDNF	(x)	x	x	x	x							multi-state disjunctive normal form (MDNF)	DNF
MIP	(x)	x	x	x	x							multi-state prime implicants	IP
d-MDNF	(x)	x	x	x	x							deterministic MDNF	d-DNF
MMODS	(x)	x	x	x	x							multi-state models	MODS
MDD	(x)	(x)	x					x	x			multivalued decision diagram (MDD)	BDD
FMDD	(x)	(x)	x					x	x			free MDD	FBDD
OMDD	(x)	(x)	x					x	(x)	x		ordered MDD (OMDD)	OBDD
$\pi$ -OMDD	(x)	(x)	x					x	(x)	(x)	x	OMDD using order $\pi$	$\pi$ -OBDD, OBDD $_{<}$





**Fig. 3.** Sub-language relationships for MDAG. An edge  $L_1 \rightarrow L_2$  indicates that  $L_1$  is a sub-language of  $L_2$ .

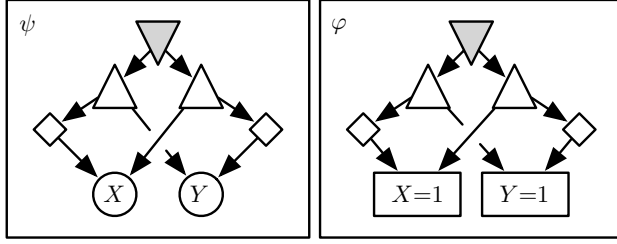
important sub-languages. Note that this table is far from being complete. We use **c**, **d**, **f**, and **n** to indicate that the properties decomposability, determinism, flatness, and no-negation hold. Figure 3 shows how the languages are related in terms of set inclusion.

## 2.2 MDAGs vs. PDAGs

In this subsection, the usage of MDAGs and PDAGs will be compared. For this purpose, we will first examine the special case of BFs, before considering the general case of finite CIFs. The reader may skip this subsection, if the interest is primarily on MDAGs.

Formally, PDAGs are almost like MDAGs, except that leaves are represented by  $\bigcirc$  and labeled with  $\top$  (true),  $\perp$  (false), or  $X$ , where  $X \in \mathbf{V}$  is a Boolean variable with  $\Omega_X = \{0, 1\}$  [6]. The language of all possible PDAGs is denoted by PDAG. The left hand side of Fig. 4 depicts a PDAG  $\psi$  with  $f_\psi = (\neg[X=1] \wedge [Y=1]) \vee ([X=1] \wedge \neg[Y=1])$ . Leaves labeled with  $\top$  ( $\perp$ ) represent the constant BF which always evaluates to 1 (0). A leaf labeled with the propositional symbol  $X$  is interpreted as the assignment  $X=1$ , i.e. it represents the BF which evaluates to 1 iff  $X = 1$ . All other nodes ( $\Delta$ ,  $\nabla$ ,  $\Diamond$ ) have the same meaning as for MDAGs.

From a PDAG  $\psi$  representing a BF  $f$ , we obtain an MDAG  $\varphi$  representing the same BF  $f$  by simply replacing  $\bigcirc$ -nodes labeled with  $X$  by  $\square$ -nodes labeled with  $X=1$ , as shown in Fig. 4. Conversely, i.e. to obtain a PDAG  $\psi$  from a MDAG  $\varphi$ ,  $\square$ -nodes labeled with  $X=1$  are replaced by  $\bigcirc$ -nodes labeled with  $X$ , and



**Fig. 4.** The BF  $f$  represented by a PDAG  $\psi$  and a MDAG  $\varphi$

$\square$ -nodes labeled with  $X=0$  are replaced by  $\diamond$ -nodes, whose children are  $\circ$ -nodes labeled with  $X$ .

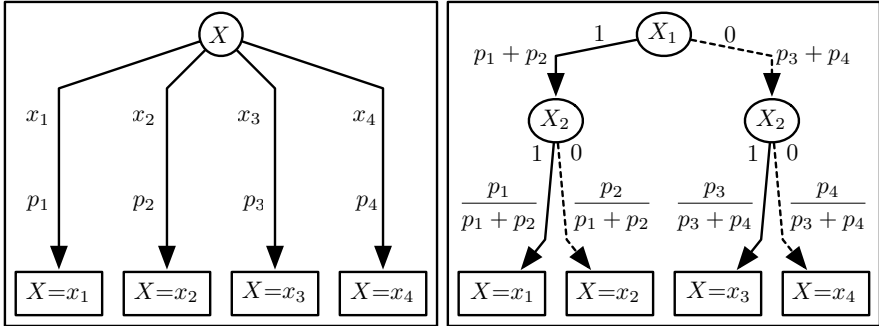
**Proposition 1.** *For every PDAG  $\psi$ , there is an equivalent MDAG  $\varphi$  with  $|\psi| = |\varphi|$ . Similarly, for every MDAG  $\varphi$  representing a Boolean function, there is an equivalent PDAG  $\psi$  with  $|\varphi| \leq |\psi| \leq |\varphi| + r$ .*

To represent CIFs by PDAGs, we need ways to transform them into BFs, i.e. each multi-state variable has to be replaced by auxiliary Boolean variables.

**Decision Diagrams:** Each multi-state variable can be replaced by  $d = \lceil \log_2 \ell \rceil$  auxiliary Boolean variables, where  $\ell$  denotes the number of possible states [10]. For each decision node in the MDD, this replacement induces a tree-shaped decision diagram of depth  $d$  and with  $\ell - 1$  binary decision nodes. Figure 5 depicts the decision diagram for the multi-state variable  $X$  with  $\Omega_X = \{x_1, x_2, x_3, x_4\}$  and its replacement with the auxiliary variables  $X_1$  and  $X_2$ , i.e. for  $d = 2$  and  $\ell = 3$ .

In [10], it is argued that this encoding corresponds to a linear transformation with a small constant. This argument is used to put the usefulness of MDDs into question. Such a conclusion is partly valid from a purely logical point of view and for small  $\ell$ , but it does no longer hold when  $\ell$  is large or when MDAGs are used to compute probabilities. In the latter case, the Boolean variables used to replace a multi-state variable are no longer independent, which disallows the classical method of probability computation. One way to overcome this is to derive respective conditional probabilities and to attach them to the edges as depicted in Fig. 5. This shows that the variable  $X_2$  depends on  $X_1$ . Since the outgoing edges of the  $X_2$  node have different probabilities, this is like using  $\ell - 1$  new variables.

**PDAGs and NNFs:** In this context, the probabilities are attached to the variables. In our example, we would have to attach two different conditional probabilities to  $X_2$ , which is impossible. An obvious alternative replacement considers each decision node of the decision diagram as an auxiliary variable, i.e.  $\ell - 1$  variables in total, where the probabilities of the variables correspond to the probabilities attached to the edges. This is essentially the replacement proposed in [13].



**Fig. 5.** Decision diagram for variable  $X$  and its replacement. The labels of the edges correspond to states and (conditional) probabilities.

An alternative replacement considers each state of a multi-state variable as a binary variable. This requires the explicit inclusion of an *exclusive or* over these auxiliary variables [11,12]. In this way, the switch to conditional probabilities is not necessary, but still the computation of probabilities becomes more difficult. A possible solution is to do some sort of *weighted model counting*, where the probabilities are attached to the leaves only, and their negations get the constant value 1.

The size of the different replacements and the additional effort strengthens our conclusion, namely that multi-state variables are useful and should be used, not encoded.

### 3 Succinctness, Queries and Transformations

The crucial properties of a language are its succinctness and the sets of queries and transformations supported in polynomial time. Depending on the application, we may come up with a set of queries and transformations, which the chosen language should support in polynomial time. If more than one language qualifies, the most succinct language provides the most compact representation. This is then the most appropriate language for the considered application.

In the following analysis of the MDAG language family, we will try to generalize as many results as possible from corresponding PDAG languages.

#### 3.1 Succinctness

With respect to two languages  $L_1$  and  $L_2$ , the intuitive idea of succinctness is to figure out whether finite CIFs are represented more compactly by elements of  $L_1$  or by elements of  $L_2$ . The following definition corresponds to the one given in [5,6].

**Definition 2.** Let  $L_1$  and  $L_2$  be two languages.  $L_1$  is equally or more succinct than  $L_2$  (or  $L_1$  is at least as succinct as  $L_2$ ), denoted by  $L_1 \preceq L_2$  iff for every

$\varphi_2 \in L_2$ , there is a  $\varphi_1 \in L_1$  such that  $\varphi_1 \equiv \varphi_2$  and  $|\varphi_1|$ , the size of  $\varphi_1$ , is polynomial in  $|\varphi_2|$ , the size of  $\varphi_2$ .

The relation  $\preceq$  is clearly *reflexive*, *anti-symmetric*, and *transitive*, i.e. it defines a *partial order* over all possible subsets of MDAG. Two languages  $L_1$  and  $L_2$  are called *equally succinct*, denoted by  $L_1 \equiv L_2$ , iff  $L_1 \preceq L_2$  and  $L_2 \preceq L_1$ . The language  $L_1$  is called *strictly more succinct* than  $L_2$ , denoted by  $L_1 \prec L_2$ , iff  $L_1 \preceq L_2$  and  $L_2 \not\preceq L_1$ . They are *incomparable*, iff  $L_1 \not\preceq L_2$  and  $L_2 \not\preceq L_1$ .

To generalize the succinctness results of PDAGs to MDAGs, let  $L_1^P$ ,  $L_2^P$  be two different PDAG sub-languages and let  $L_1^M$ ,  $L_2^M$  be their corresponding MDAG sub-languages (see Table 1). The following proposition is direct consequence of Proposition 1.

**Proposition 2.**  $L_1^P \not\preceq L_2^P \Rightarrow L_1^M \not\preceq L_2^M \quad (\equiv L_1^M \preceq L_2^M \Rightarrow L_1^P \preceq L_2^P)$ .

Proving the converse, i.e.  $L_1^P \preceq L_2^P \Rightarrow L_1^M \preceq L_2^M$ , is more difficult. Although this proof is missing in general, most of the results can be transferred, since only two methods are used to proof  $L_1^P \preceq L_2^P$ :

- Sub-language relationships: if  $L_2^P$  is a sub-language of  $L_1^P$ , then  $L_1^P \preceq L_2^P$  holds trivially. The corresponding languages  $L_1^M, L_2^M$  have of course the same sub-language relationship. Thus,  $L_1^M \preceq L_2^M$  holds.
- Providing an algorithm that obtains  $\varphi_1^P \in L_1^P$  from  $\varphi_2^P \in L_2^P$  while meeting the size restriction. Taking a closer look at these algorithms reveals that they can be adapted to multi-state variables, i.e.  $\varphi_1^M \in L_1^M$  is obtainable from  $\varphi_2^M \in L_2^M$  while meeting the size restriction. Thus,  $L_1^M \preceq L_2^M$  holds.

In this sense the succinctness relation between  $L_1^M, L_2^M$  matches the succinctness relation between  $L_1^P, L_2^P$  as given in [5,6].

### 3.2 Queries

A *query* is an operation that returns information about a MDAG representing a finite CIF without changing it. Among the important queries for finite CIFs are: *consistency* (CO) or *satisfiability* (SAT), *validity* (VA), *clause entailment* (CE), *term implication* (IM), *sentential entailment* (SE), *equivalence* (EQ), *model counting* (CT), *model enumeration* (ME), *counter-model enumeration* (ME<sup>C</sup>), *probabilistic equivalence* (PEQ), and *probability computation* (PR).

If a language supports a query in polynomial time with respect to the size of the PDAG(s)/MDAG(s) (in the case of model or counter-model enumeration, the reference size is both the size of the PDAG/MDAG and size of the satisfying set or its complement), we simply say that it *supports* this query. In the following, let  $L^M$  be a MDAG sub-language,  $L^P$  be the corresponding PDAG sub-language, and  $Q$  be a query. A direct consequence of Proposition 1 is: If  $Q$  is supported by  $L^M$ , then  $Q$  is supported by  $L^P$ . Or equivalently:

**Proposition 3.** *If  $Q$  is not supported by  $L^P$ , then  $Q$  is not supported by  $L^M$ .*

Unfortunately, the converse, i.e.  $(L^P \text{ supports } Q) \Rightarrow (L^M \text{ supports } Q)$ , is not proved in general. However, it is easy to prove it for the languages given in Table 1. If  $Q$  is supported by a language  $L$ , it is also supported by the sub-languages of  $L$ , i.e. it is enough to consider the algorithms of the super-languages. Furthermore, it is sufficient to consider  $Q \in \{\text{CO}, \text{IM}, \text{CT}, \text{EQ}, \text{SE}\}$  due to the correlations between the queries, see [6] for details.

In this sense the supported queries of  $L^M$  matches the supported queries of  $L^P$  as given in [5,6].

### 3.3 Transformations

A *transformation* is an operation that returns a MDAG representing a modified finite CIF. The new MDAG is supposed to satisfy the same properties as the language in use. Let's consider the following transformations: *term conditioning* (TC), *forgetting* (FO), *singleton forgetting* (SFO), *conjunction* (AND), *binary conjunction* (AND<sub>2</sub>), *disjunction* (OR), *binary disjunction* (OR<sub>2</sub>), and *negation* (NOT).

Note that conditioning, denoted by  $\varphi|[X=x_i]$ , includes the implicit *exclusive or*. This means the leaf labeled with  $X=x_i$  is replaced by the leaf labeled with  $\top$  and, in addition, leaves labeled with  $X=x_j$ ,  $j \neq i$ , are replaced by the leaf labeled with  $\perp$ .

If a language supports a transformation in polynomial time with respect to the size of the PDAG(s)/MDAG(s), we simply say that it *supports* this transformation. In the following, let  $L^M$  be a MDAG sub-language,  $L^P$  be the corresponding PDAG sub-language, and  $T$  be a transformation. Another direct consequence of Proposition 1 is: If  $T$  is supported by  $L^M$ , then  $T$  is supported by  $L^P$ . This is equivalent to:

**Proposition 4.** *If  $T$  is not supported by  $L^P$ , then  $T$  is not supported by  $L^M$ .*

Once more, proving the converse, i.e.  $(L^P \text{ supports } T) \Rightarrow (L^M \text{ supports } T)$ , is an open task. Nevertheless, the results can be generalized, since the proof for  $L^P$  supporting  $T$  can be adapted to  $L^M$ . Therefore, we have to consider the proof of Proposition 5.1 in [5], but only the parts where a language  $L^P$  supports a transformation  $T$ . These proofs can be extended to hold also for  $L^M$ . In this sense the set of transformations supported by  $L^M$  matches the set of transformations supported by  $L^P$  as given in [5,6].

## 4 Conclusion

By allowing multi-state variables, this paper extends the family of graph-based languages for representing BFs to the corresponding family of graph-based languages for representing finite CIFs. Our main result is the observation, that properties w.r.t. succinctness, supported queries, and supported transformation are inherited, i.e. the mostly entire knowledge compilation map is extensible from propositional to multi-state variables. This allows us to avoid the usual approach of transforming the given CIF into a BF and the resulting linear growth.

## Acknowledgments

This research supported by the *Swiss National Science Foundation*, Project No. PP002-102652/1, and *The Leverhulme Trust*.

## References

1. Akers, S.B.: Binary decision diagrams. *IEEE Transactions on Computers* **27**(6) (1978) 509–516
2. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers* **35**(8) (1986) 677–691
3. Bryant, R.E.: Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys* **24**(3) (1992) 293–318
4. Darwiche, A.: Decomposable negation normal form. *Journal of the ACM* **48**(4) (2001) 608–647
5. Darwiche, A., Marquis, P.: A knowledge compilation map. *Journal of Artificial Intelligence Research* **17** (2002) 229–264
6. Wachter, M., Haenni, R.: Propositional DAGs: a new graph-based language for representing Boolean functions. In Doherty, P., Mylopoulos, J., Welty, C., eds.: *KR'06, 10th International Conference on Principles of Knowledge Representation and Reasoning*, Lake District, U.K., AAAI Press (2006) 277–285
7. Hill, F.J., Peterson, G.R.: *Introduction to Switching Theory and Logical Design*. John Wiley and Sons, New York, USA (1974)
8. Lukasiewicz, J., Tarski, A.: Untersuchungen über den Aussagenkalkül. *Comptes rendus des séances de la Société des Sciences et des Lettres de Varsovie Cl. III* **23** (1930) 30–50
9. Rosser, J.B., Turquette, A.R.: *Many-Valued Logics*. North-Holland (1952)
10. Wegener, I.: *Branching Programs and Binary Decision Diagrams – Theory and Applications*. Number 56 in *Monographs on Discrete Mathematics and Applications*. SIAM (2000)
11. Chavira, M., Darwiche, A.: Compiling Bayesian networks with local structure. In: *IJCAI'05, 19th International Joint Conference on Artificial Intelligence*, Edinburgh, U.K. (2005)
12. Palacios, H., Bonet, B., Darwiche, A., Geffner, H.: Pruning conformant plans by counting models on compiled d-DNNF representations. In: *ICAPS'05, 15th International Conference on Planning and Scheduling*, Monterey, USA (2005) 141–150
13. Sang, T., Beame, P., Kautz, H.: Solving Bayesian networks by weighted model counting. In: *AAAI'05, 20th National Conference on Artificial Intelligence*. Volume 1., Pittsburgh, USA (2005) 475–482
14. Bollig, B., Sauerhoff, M., Sieling, D., Wegener, I.: Binary decision diagrams. In Crama, Y., Hammer, P., eds.: *Boolean Functions*. Volume II. (2006 (to appear))

# Fuzzy Clustering for Topic Analysis and Summarization of Document Collections

René Witte<sup>1</sup> and Sabine Bergler<sup>2</sup>

<sup>1</sup> Institut für Programmstrukturen und Datenorganisation (IPD)  
Universität Karlsruhe (TH), Germany

<sup>2</sup> Department of Computer Science and Software Engineering  
Concordia University, Montréal, Canada

**Abstract.** Large document collections, such as those delivered by Internet search engines, are difficult and time-consuming for users to read and analyse. The detection of common and distinctive topics within a document set, together with the generation of multi-document summaries, can greatly ease the burden of information management. We show how this can be achieved with a clustering algorithm based on fuzzy set theory, which (i) is easy to implement and integrate into a personal information system, (ii) generates a highly flexible data structure for topic analysis and summarization, and (iii) also delivers excellent performance.

## 1 Introduction

Information seekers nowadays are typically overwhelmed with the multitude of documents available for any given topic online. Whereas information retrieval (IR) is adequately covered by modern Internet search engines, the user is mostly left alone with the task of sifting through the resulting set of documents. Delivering automated tools for the analysis, filtering, and pre-processing of natural language texts is an important next step in personal information management.

In this paper, we provide a fuzzy clustering algorithm for the analysis of document collections, as they could have resulted from a query posed to an Internet search engine or an intranet document server. We show how both common and distinctive topics of such a document set can be detected, which is far more informative than simple keyword extracts. Additionally, the data structure computed by our algorithm allows for the generation of several types of multi-document summaries, including a context-sensitive one. Such summaries, typically between 100 and 350 words, are more easily scanned by a human; cross-linked with the original documents they further aid a user in quickly determining relevant topics and help in deciding which documents are a good candidate to read in full.

Our research is significant for several reasons: (1) We provide a flexible, adaptive, context-sensitive algorithm for the analysis of natural language document collections, which is easy to implement and integrate into information systems, thereby substantially improving information management for users; (2) We show how fuzzy set theory can be applied to natural language processing (NLP), which adds robustness, ease of deployment, and flexibility to our approach, while at

the same time allowing for further transfer of ideas and algorithms from the field of soft computing to information management; (3) We show how to generate a highly flexible data structure that can be used to generate multiples types of condensed information displays that can also adapt to a user's current context; and (4) Results from large-scale evaluations on data from the international *Document Understanding Conference* (DUC) competition prove our approach to be highly competitive with current summarization systems.

The remainder of this paper is structured as follows: in the next section, we describe the preprocessing needed to provide the input required by our algorithm. Section 3 describes our contribution, the cluster algorithm. Some possible applications of the generated data structure, including topic detection and summarization, are outlined in Section 4. An evaluation of our approach, based on data and methods from the NIST-sponsored DUC competition, is presented in Section 5. Section 6 discusses related work, followed by conclusions in Section 7.

## 2 Preprocessing

Before we can describe our main fuzzy clustering technique, we have to discuss some preprocessing steps needed to generate the required input data structures: noun phrase (NP) chunks and fuzzy coreference chains. Additionally, this section provides a first introduction to the idea of applying fuzzy set theory [1] to natural language processing.

### 2.1 Noun Phrases

All documents first undergo basic NLP preprocessing, including tokenization, sentence splitting, and part-of-speech (POS) tagging, which in our implementation is performed within the open source GATE (*General Architecture for Text Engineering*) framework.<sup>1</sup>

Noun phrases, i.e., determiner/modifier/head triples, are then computed as the main input to the following coreference resolution step. Here, we assume minimal (base) NPs, without prepositional or other attachments. Our NPs are computed by the MuNPEx chunker<sup>2</sup> based on part-of-speech tags and additional entity-specific grammar rules. However, since for the purpose of cluster-based coreference resolution it does not matter how NPs have been computed, an algorithm based on a full parse would work equally well.

### 2.2 Fuzzy Coreferences

Our technique is based on grouping NPs into *fuzzy coreference chains*. Our approach for coreference resolution is based on fuzzy set theory as the underlying

<sup>1</sup> For more detail on these steps, we refer the reader to the GATE user's guide at <http://gate.ac.uk/sale/tao/index.html>

<sup>2</sup> Multi-lingual Noun Phrase Extractor (MuNPEx), <http://www.ipd.uka.de/~durm/tm/munpex>

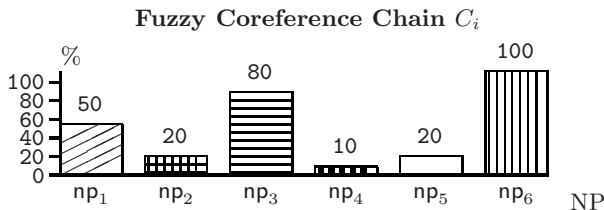


formal representation. For the purpose of this paper, we will only give a brief outline of fuzzy coreference resolution. A detailed description of our algorithm is available in [2, 3].<sup>3</sup>

Fuzzy coreference resolution groups entities (typically NPs) into fuzzy coreference chains. Each chain contains all textual descriptions that refer to the same entity (e.g., in a text, the three descriptions “*Luke Skywalker*,” “*he*,” and “*the young Jedi*” might refer to the same person). The central idea behind using fuzzy set theory is the uncertainty inherent in natural language processing: even for a human reader, it is not always certain whether two NPs really refer to the same entity; employing fuzzy sets allows a soft computing approach where this uncertainty is represented explicitly, rather than making decisions based on (often arbitrary) hard thresholds.

**Fuzzy Coreference Chains.** Fuzzy coreference chains link entities, which are typically represented by noun phrases (NPs). In this paper, we denote the set of all noun phrases within a text with the (crisp) set  $NP = \{np_1, \dots, np_m\}$ , i.e., there are  $m$  noun phrases within a document. A single fuzzy chain  $C$  is then represented by a fuzzy set  $\mu_C$ , which maps the domain of all noun phrases  $NP$  to the  $[0, 1]$ -interval:  $\mu_C : NP \rightarrow [0, 1]$ . Thus, each noun phrase  $np_i \in NP$  has a membership degree  $\mu_C(np_i)$ , indicating how certain this NP is a member of chain  $C$ . The membership degree for a single noun phrase  $\mu_C(np_i) \in [0, 1]$  is interpreted in a possibilistic fashion: a value of 0.0 (“*impossible*”) indicates that the NP cannot be a member of chain  $C$ , a value of 1.0 (“*100% possibility*” or “*certain*”) means that none of the available information indicates that the NP is not in the chain, intermediate values represent different degrees of compatibility of a noun phrase with the chain.

*Example (fuzzy coreference chain).* Fig. 1 shows an example for a fuzzy coreference chain  $C_i$ . Here, the noun phrases  $np_3$  and  $np_6$  have a very high possibility for belonging to the chain,  $np_1$  only a medium possibility, and the remaining NPs are most likely not chain members.



**Fig. 1.** Fuzzy chain  $C_i$  with membership grades for each noun phrase

The output of a fuzzy coreference algorithm is a set of fuzzy coreference chains, similarly to classical coreference resolution systems. Each chain holds all noun

<sup>3</sup> Additionally, [4] provides an overview of different coreference algorithms, including an independent implementation and evaluation of our approach (albeit not using fuzzy sets, i.e., a crisp implementation).

phrases that refer to the same conceptual entity. However, unlike for classical, crisp chains, we do not have to reject inconsistent information out of hand, so we can admit a noun phrase as a member of more than one chain, with different degrees of certainty for each. This provides an explicit representation of the uncertainty that is so common in natural language analysis.

Fuzzy chains can be converted to crisp chains using a *defuzzification* function, which allows downstream language analysis components that are not fuzzy-aware to use results of a fuzzy algorithm.

**Fuzzy Coreference Resolution.** Fuzzy chains are constructed through (usually knowledge-poor) *fuzzy heuristics*. Typical features used within our heuristics are *head noun*, *gender*, or *position*.

Our fuzzy coreference algorithm is essentially a single-link hierarchical clustering strategy. It initially creates one fuzzy chain for each NP, which forms its medoid (for example, in Fig. 1,  $np_6$  is the chain's medoid). We then compute the degree of coreference between all NP pairs within a text, each degree normalized to a fuzzy value in the  $[0, 1]$ -interval. Each fuzzy degree can be interpreted as a distance between the medoid and the co-referring NP; it is added to every chain using standard fuzzy set operators. For example, in Fig. 1, at least one fuzzy heuristic must have determined a fuzzy coreference degree of 0.8 for  $(np_6, np_3)$ .

Finally, all chains are *merged* using a prescribed consistency degree  $\gamma$ . Merging combines compatible chains into merged chains (or NP clusters) using the coreference properties of symmetry and transitivity. The merge degree  $\gamma$  influences the size of the chains, and in effect, their precision and recall. A degree of 0 would merge all NPs into a single (yet useless) chain, while a value of 1 would lead to chains of the best possible precision, leaving out uncertain links and thereby resulting in more singletons (and lower recall).

The process of merging is now repeated for each possible value of  $\gamma \in \{\gamma_1, \dots, \gamma_n\}$ ,<sup>4</sup> leading to a *family* of coreference chains, a set of sets of chains:  $\mathcal{C} = \{C_1^{\gamma_1}, \dots, C_n^{\gamma_n}\}$ . Note that a similar result can be obtained with a non-fuzzy coreference clustering strategy.

For the purpose of our algorithm described in the next section it is important that the individual chains exhibit *monotonicity*, that is, if two entities are linked within a chain of a specific certainty  $\gamma_i$ , they must also be linked in all chains of lower certainty  $\gamma_j \leq \gamma_i$ .

We use the same algorithm to create both inter- and intra-document coreference chains, only the number of enabled heuristics and various parameters differ for each. The end results are two families of coreference chains, one for intra- and one for inter-document coreferences.<sup>5</sup>

<sup>4</sup> Since fuzzy sets are stored in horizontal representation through a set of  $\alpha$ -cuts, with  $[\mu]_\alpha = \{\omega \in \Omega \mid \mu(\omega) \geq \alpha\}$ , the merge degree  $\gamma$  can only assume a finite number of different values, which typically correspond to the  $\alpha$ -cut levels (e.g.,  $\alpha \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ ).

<sup>5</sup> Cross-document chains do not contain links between NPs of the same document, since these links have already been computed by the intra-document step.

### 3 Fuzzy Coreference Graph Clustering

In this section, we describe our main contribution, a fuzzy coreference cluster graph algorithm that builds the data structure needed for identifying topics in document sets and constructing various types of summaries. This algorithm takes as input the intra- and inter-document coreference chain families computed by a coreference algorithm under different (fuzzy) clustering thresholds as described in the previous section.

The first step is the construction of an initial fuzzy coreference cluster graph, as described in Section 3.1 below. Our clustering algorithm, described in Section 3.2, then works on this data structure, computing clusters that can be used to create several kinds of condensed information displays, including multi-document summaries (Section 4).

Essentially, two kinds of clusters are created by our algorithm: *common* clusters that contain NPs from two or more documents, and *distinctive* clusters that contain NPs from one document only. In other words, each cluster determines a *topic* that can be tracked through the different documents: some topics span all documents (common topic), some topics only occur in a subset or a single document (contrastive/distinctive topic).

#### 3.1 Cluster Graph Initialization

A *fuzzy coreference cluster graph* is an undirected, weighted graph with entities (typically NPs) as nodes and weighted coreferences between these entities as edges. Essentially, it folds both inter- and intra-document coreference chains into one data structure that can then be traversed by the clustering algorithm. Thus, the algorithm's input are the intra- and inter-document coreference families, computed by one of the standard coreference clustering algorithms:

*Input (cluster graph initialization).* Input to the cluster initialization step is a set of sets of coreference chains  $\mathcal{C} = \mathcal{C}_{\text{inter}} \cup \mathcal{C}_{\text{intra}}$  with the inter-document chains  $\mathcal{C}_{\text{inter}} = \{C_1^{\gamma_1}, \dots, C_n^{\gamma_n}\}$  and the intra-document chains  $\mathcal{C}_{\text{intra}} = \{C_{n+1}^{\gamma_1}, \dots, C_{2n}^{\gamma_n}\}$ .

Note that each coreference chain  $C_i^\gamma$  contains again a set of sets of NPs, where all NPs within a subset  $c \in C$  corefer with a fuzzy certainty degree of  $\gamma$ . We can now create the initial cluster graph.

*Definition (initial cluster graph).* An initial cluster graph  $\mathcal{G} = (V, E)$  is constructed from the intra- and inter-document coreference families as follows. The set of graph nodes  $V$  is given by the set containing all NPs from all documents. The set of edges is derived from the set  $\mathcal{C}$  containing both intra- and inter-document coreference families by iterating through all coreferences  $C \in \mathcal{C}$ . For each chain  $c \in C$ , we then iterate through all the entities  $(np_i, np_j)$  within that chain and create one edge of weight  $\gamma$  between them. The complete algorithm is shown in Fig. 2. Note that we treat coreferences as links, that is, for a coreference chain  $c_i^\gamma = \{np_1, np_2, np_3\}$  we add two edges with weight  $\gamma$  to the graph, one between  $np_1$  and  $np_2$  and one between  $np_2$  and  $np_3$ .

**Require:** Set of coreference families  $\mathcal{C}$ ,  
graph  $\mathcal{G}$

```

1: for all  $C \in \mathcal{C}$  do
2:   for all  $c \in C$  do
3:     for  $i=1$  to  $|c| - 1$  do
4:        $np_i \leftarrow c.get(i);$ 
5:        $np_j \leftarrow c.get(i+1);$ 
6:       if  $e=(np_i, np_j) \in E$  then
7:          $\gamma' \leftarrow \max(C_\gamma, e_\gamma);$ 
8:          $updateEdge(np_i, np_j, \gamma');$ 
9:       else
10:         $addEdge(np_i, np_j, C_\gamma);$ 

```

Fig. 2. Cluster Graph Initialization

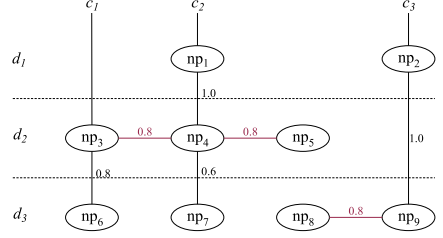


Fig. 3. Initialized fuzzy cluster graph

*Example (initial cluster graph).* Consider three documents  $d_1, d_2, d_3$  with two coreference families (inter- and intra-document), containing three coreference sets each for  $\gamma \in \{0.6, 0.8, 1.0\}$ :<sup>6</sup>

$$\mathcal{C}_{\text{inter}} = \{C_1, C_2, C_3\}, \mathcal{C}_{\text{intra}} = \{C_4, C_5, C_6\}$$

With the inter-document chains  $C_1, C_2, C_3$ :

$$\begin{aligned} C_1 &= \{\{np_3, np_6\}, \{np_1, np_4, np_7\}, \{np_2, np_9\}\} \\ C_2 &= \{\{np_3, np_6\}, \{np_1, np_4\}, \{np_2, np_9\}\} \\ C_3 &= \{\{np_1, np_4\}, \{np_2, np_9\}\} \end{aligned}$$

and the intra-document chains  $C_4, C_5, C_6$ :

$$\begin{aligned} C_4 &= \{\{np_3, np_4, np_5\}, \{np_8, np_9\}\} \\ C_5 &= \{\{np_3, np_4, np_5\}, \{np_8, np_9\}\} \\ C_6 &= \{\} \end{aligned}$$

Fig. 3 shows the resulting initial cluster graph. Intra-document coreference chains are drawn horizontally (in red), while cross-document chains (in black) are displayed from top to bottom. Each edge in the graph is labeled with the fuzzy *certainty* value of the coreference; in the example,  $np_4$  and  $np_5$  corefer with a certainty of 0.8, while  $np_4$  and  $np_7$  corefer with a certainty of 0.6.

### 3.2 The Clustering Algorithm

We can now describe the main clustering algorithm that works on the initial data structure described above. Similarly to chain merging, graph clustering is controlled by a threshold  $\theta$ . In general, the lower the clustering threshold, the more entities are clustered together, resulting in fewer, but larger, clusters.

The key idea is to use the degree of coreference between entities, represented by an edge's weight, as the *inverse distance* between those entities: entities linked by an edge of weight 1.0 are closest, whereas entities with an edge of weight 0.0 (i.e., no edge) are infinitely far apart. We can now apply an agglomerative hierarchical clustering strategy, creating a dendrogram data structure.

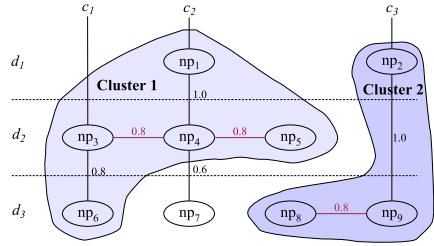
<sup>6</sup> Singletons are omitted for brevity.

**Require:** Initialized cluster graph  $\mathcal{G} = (V, E)$

```

1: for all  $np \in V$  do
2:   createCluster(  $np$  ); {create initial clusters}
3: for all  $e \in E$  do
4:   if  $e_\gamma \geq \theta$  then {join clusters?}
5:      $c_1 \leftarrow$  getCluster(  $e_{start}$  );
6:      $c_2 \leftarrow$  getCluster(  $e_{end}$  );
7:     mergeClusters(  $c_1, c_2$  );
```

**Fig. 4.** Fuzzy Clustering Algorithm



**Fig. 5.** Graph after running the clustering algorithm with  $\theta = 0.8$

*Definition (coreference graph clustering).* The clustering process starts with clusters containing individual entities, i.e., each node  $V$  in the initialized graph  $\mathcal{G}$  represents a cluster by itself. We now apply a hierarchical clustering strategy, where we progressively merge clusters until the algorithm terminates. Two clusters are merged if a direct edge exists between them of weight  $\gamma \geq \theta$ . If multiple edges exist between two clusters, we evaluate the one with the highest weight, i.e., we use a single-linkage clustering strategy. The algorithm terminates when no more edges exist between clusters. Fig. 4 shows the complete algorithm.

When the cluster algorithm terminates, the cluster graph contains clusters spanning multiple documents (common topics), as well as clusters that contain entities from a single document (distinctive topics). Of course, either set may be empty, depending on the input document set and the threshold setting.

*Example (final cluster graph).* Fig. 5 shows the result after running the clustering algorithm on the graph in Fig. 3 with  $\theta = 0.8$ . This results in two large common NP clusters and the distinctive topic  $np_7$  in document  $d_3$  (documents  $d_1$  and  $d_2$  do not have any distinctive topics). For  $\theta = 0.6$ , however,  $np_7$  would have been added to cluster 1, whereas a larger  $\theta$  value would have created smaller clusters and more singletons.

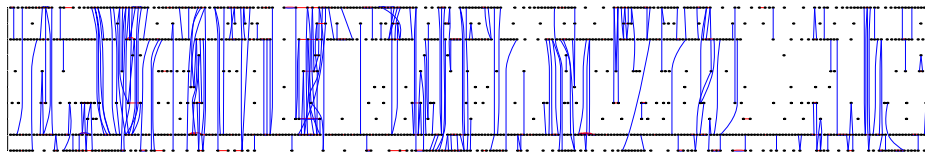
Note that we can repeat the clustering process for each fuzzy value of  $\theta$ , which results in a cluster family (or one multi-dimensional cluster). However, within this paper, we will only discuss single clusters.

## 4 Topic Analysis and Automatic Summarization

In this section, we discuss a number of applications for our cluster graph data structure for the analysis of document collections, including topic detection and multi-document summarization.

### 4.1 Common Topic Detection

The first application we discuss is the detection of a *common topic* within a collection of documents. This allows a user to easily identify the most salient information with a set of texts.



**Fig. 6.** Initial cluster graph for a DUC ten-document set: 2779 nodes (NPs), 3822 edges (co-references). Intra-document links (red) are aligned horizontally and cross-document links (blue) vertically (enlarge electronic version for details).

Common clusters
Hurricane Mitch in Central America (31) – Honduras (21) – the country’s central coast (15) – last week’s storm (12)

**Fig. 7.** Common topic clusters for a set of ten documents on “Hurricane Mitch” (number of NPs for common clusters in parentheses)

This kind of information can be immediately generated from the cluster graph data structure: Topics are identified by clusters, so by extracting clusters that span all documents (or a sufficiently large subset thereof), a system can obtain the common themes of all documents. In order to rank the topics by relevance, the *size* of each cluster can be additionally evaluated: the larger a cluster, the more important the topic contained within (we give empirical evidence for this in Section 5).

*Example (common topic detection).* We give a real-world example based on the DUC 2004 [5] data set **d30002**, which contains ten documents discussing *Hurricane Mitch*. Fig. 6 shows the initial cluster graph generated for this document set; after clustering, the common topics can be identified as shown in Fig. 7.

As can be seen, for a specific clustering threshold, four topic clusters have been identified, which are addressed in all ten documents. Changing the fuzzy threshold results in different clusters, either more discriminative (more and smaller clusters) for larger  $\theta$  values or more lenient (fewer and larger clusters). However, we cannot show these here due to space constraints.

## 4.2 Multi-document Summarization

The list of common topic identifiers presented in Fig. 7 provides a highly condensed view of a document collection. If a user is interested in obtaining more detail, but without reading each of the documents involved, he may opt to view a *multi-document summary*.

Multi-document summarization attempts to identify the most salient (shared) topics within a collection of documents. The summary, typically sentences (or sentence parts) extracted from the documents, should reflect as many common topics as space permits. Current systems (see the DUC Proceedings [6, 5]) typically achieve this by ranking sentences within a document collection, either through statistical means or (shallow) linguistic analysis and subsequent relevance scoring.

Common Topic Summary
The Honduran president closed schools and public offices on the coast Monday and ordered all air force planes and helicopters to evacuate people from the Islas de la Bahía, a string of small islands off the country's central coast. National police spokesman Ivan Mejia said the Coco, Segovia and Cruta rivers all overflowed their banks Monday along Honduras' eastern coast. The European Union on Tuesday approved 6.4 million European currency units (dlrs 7.7 million) in aid for thousands of victims of the devastation caused by Hurricane Mitch in Central America. The greatest losses were reported in Honduras, where an estimated 5,000 people died and 600,000 people – 10 percent of the population – were forced to flee their homes after last week's storm.

**Fig. 8.** Cluster graph generated multi-document summary

As shown above, our cluster graph data structure already represents common topics in a document set. We implemented a summarization component that generates a multi-document summary by selecting (at least) one candidate noun phrase from each cluster, in decreasing order of importance (cluster size), until a prescribed length limit has been reached or all clusters are exhausted. The candidate NPs, in turn, can be used to select the sentences they appear in as a candidate text extract.

*Example (multi-document summary).* Fig. 8 shows an example for a (roughly) 100-word summary generated with our approach [7, 8].

Extractive sentence-based summarization typically involves additional techniques, i.e., replacing dangling pronominal references, eliminating duplicate noun phrases, or removing relative clauses. However, within the scope of this paper we are not concerned with this kind of post-processing, which is already widely discussed in the literature (see e.g. [9] and the DUC proceedings).

### 4.3 Differential Topic Analysis

So far, we concentrated on the most important, common topics within a document collection. Often, a user is also interested in what *additional*, unique information a document contains, i.e., knowledge that cannot be found in other texts.

This requires a differential topic analysis. For this, we inspect the distinctive clusters generated by our algorithm. These are clusters that span only a single document, or a (configurable) subset of all texts in a set. Each of these clusters represents a topic of a document subset that is distinctive from common topics across all documents.

*Example (differential topic analysis).* Fig. 9 shows an example for a differential topic analysis, on the same document set as before. For each document, a representative phrase from each of its distinctive clusters is displayed. Note that for  $D_1$ – $D_3$  and  $D_5$ – $D_7$ , these topics identify a country that is distinctively discussed in this document only.  $D_9$  quite clearly focuses on the ensuing medical emergency. Within a user interface, these clusters could again be hyperlinked to expand into a summary of the document set, similarly to the common topic summary discussed above.

### 4.4 Context-Based Summarization

The last type of application we address here are *focused* summaries, which are not concerned with summarizing a document (set), but rather with collecting

Common clusters	
Hurricane Mitch in Central America (31) – Honduras (21) – the country’s central coast (15) – last week’s storm (12)	
Distinctive clusters	
$D_1$	Gen. Mario Hung Pacheco – the shelves of some stores and some gasoline stations – mayor of Utila – a hurricane warning – the northwest Caribbean for five days
$D_2$	the western Caribbean on Wednesday – 165 kms – Honduras with 120 – west at only 2 mph – a resident of Guanaja Island
$D_3$	the center – emergency measures on the Caribbean coast of the Yucatan Peninsula – a boat – hotels – The storm’s power
$D_4$	the storm’s death toll in the region to 357 – 231 people have been confirmed dead
$D_5$	floods – the Guatemalan border – a state of emergency – 50 kph – late Sunday
$D_6$	area – the slopes of the Casita volcano in northern Nicaragua – Sunday night – a 32-square mile – addition
$D_7$	homes – The greatest losses – affiliate in San Miguel province – a statement – the EU
$D_8$	the audience – all public and private institutions and all men – the pope – a gift – six Russian cosmonauts
$D_9$	access to places – other countries – the recovery effort – More help – at least 300 children at the shelter for diarrhea, conjunctivitis and bacterial infections
$D_{10}$	Taiwan – aid and pledges of assistance – Residents – Cuba’s offer – the saddest thing

**Fig. 9.** Differential topic analysis results based on the cluster graph generated for a set of ten documents on the “Hurricane Mitch” topic

“Who is Stephen Hawking?”
Hawking, 56, is the Lucasian Professor of Mathematics at Cambridge, a post once held by Sir Isaac Newton. Hawking, 56, suffers from Lou Gehrig’s Disease, which affects his motor skills, and speaks by touching a computer screen that translates his words through an electronic synthesizers. Stephen Hawking, the Cambridge University physicist, is renowned for his brains. Hawking, a professor of physics and mathematics at Cambridge University in England, has gained immense celebrity, written a best-selling book, fathered three children, and done a huge amount for the public image of disability. Hawking, Mr. Big Bang Theory, has devoted his life to solving the mystery of how the universe started and where it’s headed.

**Fig. 10.** Focused summary of ten documents based on a question, generated from a cluster graph

information on an explicit interest expressed through context information. For example, a user might work on a report and needs information concerning a number of questions; or he might write an email reply and would like to see existing information relevant to the emails he is answering.

Indeed, the cluster graph also allows us to generate focused summaries by including the context information as an additional document  $D_0$  when creating and clustering the fuzzy coreference chains. Then, all clusters that overlap with document  $D_0$  also contain information relevant to this context. All other clusters, even if they are bigger, are discarded for this kind of summary.

*Example (differential topic analysis).* An example for a context-based summary is shown in Fig. 10. Here, the user stated an explicit question “*Who is Stephen Hawking?*”. The focused summary is then generated from a document set using the cluster graph data structure as discussed above. As before, elements within the clusters have to be further ranked, extracted, and post-processed to create the final summary.



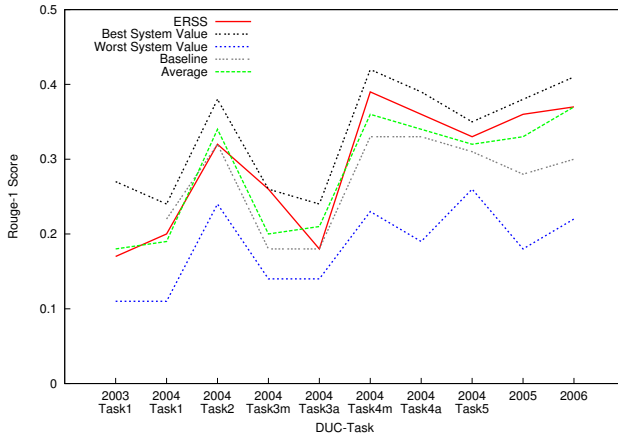


Fig. 11. Performance of the cluster graph algorithm on the DUC data 2003–2006

## 5 Evaluation

We evaluated our fuzzy coreference cluster graph algorithm using the data from the DUC competition on summarization. This involved both multi-document summaries (2003/2004) and focused summaries (2005/2006) of sets between 10 and 50 documents in size. To allow a comparison with all other systems that participated in the competition, we computed the ROUGE-1 score [10] for all years. The results of our system ERSS, compared with the best, worst, baseline, and average system of each year and task is shown in Fig. 11.

In general, our algorithm performs well above the average, in some cases within a statistically insignificant difference from the top system. More detailed results, including experiments with different fuzzy values, different evaluation measures like Basic Elements (BE), as well as a comparison of our cluster algorithm with baseline ranking strategies like  $TF*IDF$ ,<sup>7</sup> can be found in [7, 8].

## 6 Related Work and Discussion

In [11] the authors define the problem of “comparative text mining” (CTM) for a given text collection as “(1) discovering the different common themes across all the collections; (2) for each discovered theme, characterize what is in common among all the collections and what is unique to each collection.” They also apply a clustering strategy based on a cross-collection mixture model, but using only simple word-level statistics, which we believe is much less useful for creating summaries than our entity-based clustering approach.<sup>8</sup>

<sup>7</sup> The results obtained through the cluster graph significantly outperform  $TF*IDF$ -based ranking.

<sup>8</sup> A typical example cluster in [11] is the topic list “*port, jack, ports, will, your, warm, keep, down*”.

Similar work is done in the *Web Mining* community, however, they are more concerned with the static and dynamic structure of web pages (i.e., inbound/outbound links), making work like [12] more suited to information retrieval than summarization. The research area of *change summarization* is concerned with tracking a single document (or a document collection) over time and extracting new/fading topics. [13] evaluate such changes, providing the result in form of web page ranking lists.

Clustering approaches have long been applied to document analysis (see e.g. [14] for an overview), including summarization (e.g., [15]), but our work differs in that we cluster *coreferences* rather than individual (TF\*IDF-weighted) words.

## 7 Summary and Conclusions

Delivering tools for the automated analysis, structuring, and compression of information contained in natural language texts is an important need of end users overwhelmed with the task of manually filtering through search engine results. For the system engineer developing appropriate solutions, features such as robustness, flexibility, context-sensitivity and adaptability are essential properties. In this paper, we present the fuzzy coreference cluster graph algorithm as a possible solution that exhibits robustness and adaptability due to its reliance on fuzzy sets. It creates the highly flexible cluster graph data structure, which can also be employed for context-sensitive information filtering. It is also easy to implement and outperforms many existing summarization systems, most of which are in addition highly specific to a single task.

More work is needed on integrating information analysis and summarization algorithms, such as ours, into the desktop environments of knowledge workers. We present some ideas on this in [16], where the algorithm proposed here is implemented within a semantic desktop for building historians and architects analysing a 19th century encyclopedia written in German.

## References

1. Klir, G.J., Folger, T.A.: Fuzzy Sets, Uncertainty, and Information. Prentice-Hall (1988)
2. Witte, R.: Architektur von Fuzzy-Informationssystemen. BoD (2002) ISBN 3-8311-4149-5.
3. Witte, R., Bergler, S.: Fuzzy Coreference Resolution for Summarization. In: Proceedings of 2003 International Symposium on Reference Resolution and Its Applications to Question Answering and Summarization (ARQAS), Venice, Italy, Università Ca' Foscari (June 23–24 2003) 43–50 <http://rene-witte.net>.
4. Angheluta, R., Jeuniaux, P., Mitra, R., Moens, M.F.: Clustering Algorithms for Noun Phrase Coreference Resolution. In: Proc. of 7èmes Journées internationales d'Analyse statistique des Données Textuelles, Louvain La Neuve, Belgium (March 10–12 2004) 60–70
5. DUC 2004 Workshop on Text Summarization, Boston Park Plaza Hotel and Towers, Boston, USA (May 6–7 2004) NIST. <http://duc.nist.gov/pubs.html#2004>.

6. Proceedings of the HLT/NAACL Workshop on Text Summarization, Edmonton, Canada (May 31–June 1 2003) NIST.
7. Witte, R., Krestel, R., Bergler, S.: ERSS 2005: Coreference-Based Summarization Reloaded. In: Proceedings of Document Understanding Workshop (DUC), Vancouver, B.C., Canada (October 9–10 2005)
8. Witte, R., Krestel, R., Bergler, S.: Context-based Multi-Document Summarization using Fuzzy Coreference Cluster Graphs. In: Proceedings of Document Understanding Workshop (DUC), New York City, NY, USA (June 8–9 2006)
9. Mani, I.: Automatic Summarization. John Benjamins B.V. (2001)
10. Lin, C.Y.: ROUGE: a Package for Automatic Evaluation of Summaries. In: Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), Barcelona, Spain (July 25–26 2004)
11. Zhai, C., Velivelli, A., Yu, B.: A Cross-Collection Mixture Model for Comparative Text Mining. In: Proc. of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'04), ACM Press (2004) 743–748
12. Liu, B., Ma, Y., Yu, P.S.: Discovering unexpected information from your competitors' web sites. In: Knowledge Discovery and Data Mining. (2001)
13. Jatowt, A., Bun, K.K., Ishizuka, M.: Change Summarization in Web Collections. In: Innovations in Applied Artificial Intelligence: 17th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. LNCS (2004) 653–662
14. Berry, M.W.: Survey of Text Mining: Clustering, Classification, and Retrieval. Springer (2003)
15. Radev, D.R., Jing, H., Styś, M., Tam, D.: Centroid-based summarization of multiple documents. *Inf. Process. Manage.* **40**(6) (2004) 919–938
16. Witte, R., Gerlach, P., Joachim, M., Kappler, T., Krestel, R., Perera, P.: Engineering a Semantic Desktop for Building Historians and Architects. In: Proceedings of the Semantic Desktop Workshop at the ISWC. Volume 175 of CEUR Workshop Proceedings., Galway, Ireland (November 6 2005) 138–152

# Creating a Fuzzy Believer to Model Human Newspaper Readers

Ralf Krestel<sup>1</sup>, René Witte<sup>1</sup>, and Sabine Bergler<sup>2</sup>

<sup>1</sup> Institut für Programmstrukturen und Datenorganisation (IPD)  
Universität Karlsruhe (TH), Germany

<sup>2</sup> Department of Computer Science and Software Engineering  
Concordia University, Montréal, Canada

**Abstract.** We present a system capable of modeling human newspaper readers. It is based on the extraction of reported speech, which is subsequently converted into a fuzzy theory-based representation of single statements. A domain analysis then assigns statements to topics. A number of fuzzy set operators, including fuzzy belief revision, are applied to model different belief strategies. At the end, our system holds certain beliefs while rejecting others.

## 1 Introduction

With the huge success of the internet, the natural language processing (NLP) research community has developed whole branches that deal explicitly with vast amounts of unstructured information encoded in written natural language. One goal is to gain knowledge about hard facts like “The number of inhabitants of city *X*” or the “name of the president of country *Y*.” But a lot of information, especially within newspaper articles, are not hard facts, which could be easily proven right or wrong. Often newspaper articles contain different *views* of the same event, or state controversial opinions about a certain topic. In this case the notion of *belief* becomes relevant.

For humans, this is a daily task. Depending on context information and background knowledge, together with other belief structures, humans tend to believe certain statements while other statements are rejected. The process of believing also varies between different humans, not only depending on their different background knowledge, but also on different attitudes towards a coherent worldview or importance and their ability of logic thinking.

For a computational system simulating a human newspaper reader by imitating his belief processing, this involves not only the extraction of beliefs stated in an article, but also their comparison to existing beliefs held by the system. Such an *artificial believer* [1] must have different belief strategies to model different human approaches.

The application area of an artificial believer is large. Potential users include:

- Companies interested in customers’ opinions about their own products or products from a competitor.

- Governments interested in the opinions of people about their country or the government’s work.
- Individuals, who wish to have a personalized news digest compiled automatically.

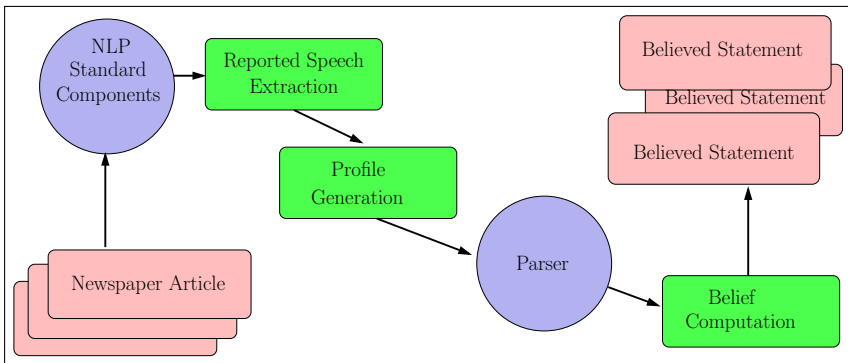
Our system is designed for the last group of users, but is not limited to this application.

The system we present in this paper addresses various problems within the NLP domain. Our main contributions are: 1. Developing rules to identify and extract reported speech from newspaper articles; 2. processing the gained information by applying fuzzy set theory to natural language processing; 3. creating a working implementation of these ideas, together with an evaluation environment.

The remainder of this paper is structured as follows: In the next section, we give an overview of our fuzzy believer system, followed by a more detailed description of the individual components in Section 2. An evaluation of our approach, using different corpora and evaluation methods, is presented in Section 3. Section 4 discusses related work, followed by conclusions in Section 5.

## 2 Design and Implementation

The core concept embodied in our approach is the application of fuzzy set theory to the NLP domain. This allows for an explicit modeling of fuzziness inherent to natural languages and enables the user to control the system’s behaviour by varying various runtime parameters responsible for the fuzzy processing. Reported speech statements present the basic set of beliefs for our system. These kinds of statements usually express a belief held by the source of the statement and allows a clear attribution of the statement to this source. The extracted reported speech structures are further processed and the output of external semantic parsers is utilized to identify predicate-argument structures (PAS) within the reported speech content. Each PAS defines a statement, which the system



**Fig. 1.** Fuzzy Believer System Architecture Overview

eventually either believes or rejects. They also form the foundation for the fuzzy processing and the basis for our heuristics to process beliefs.

To mirror the different processing steps, our fuzzy believer system consists of a set of components running consecutively. It is implemented using GATE (General Architecture for Text Engineering) [2], which offers a framework for developing NLP applications. For preprocessing, we use a number of standard components shipped with GATE, for high-level processing we developed our own components. An overview of the system's structure is shown in Figure 1, indicating the four main components constituting our system: (1) Reported speech extraction; (2) Profile generation; (3) PAS extraction; (4) Belief computation.

## 2.1 Previous Work

A similar system extracting reported speech from newspaper articles together with its source and reporting verb is presented in [3] and [4]. The system passes the extracted information through evidential analysis and processes the results to different *profiles*.

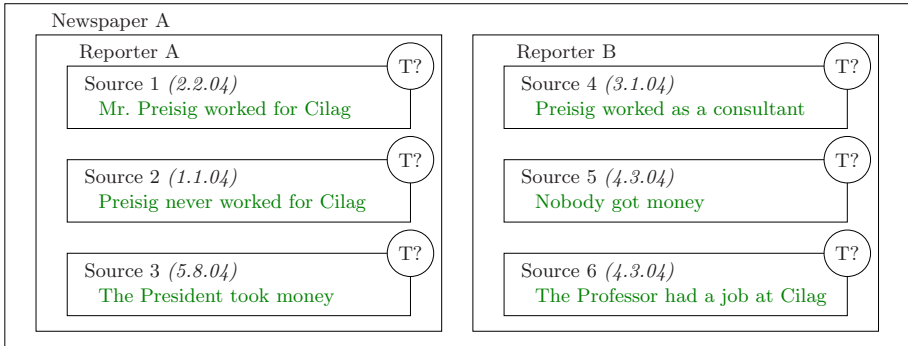
In detail, to evolve profiles out of *basic profiles*, which consist of a statement and its source, an intermediate step (*merged profiles*) is needed. In this step, the exploitation of coreference information becomes necessary. For this reason, a noun phrase coreferencer [5] is used to identify same sources of different statements. These statements are then merged into a single merged profile.

Evidential chains are generated and a percolation algorithm is used, see [6]. The merged statements are grouped according to the reporter who uttered the reported speech. This allows to model different degrees of confidence into a certain newspaper, a certain reporter, and a certain source. To encode the different confidences in the resulting profile, a dichotomy of held beliefs and potential beliefs is introduced.

In contrast to our fuzzy believer, this system is limited to handling beliefs without considering their content, solely based on information about the source of a reported clause and the reporter of the article. Apart from improving the *extraction of reported speech*, the system presented in this paper is capable of *identifying the topic* of the reported speech and for each topic the *polarity of individual statements* concerning the topic. On top of this information, an *artificial believer* is implemented simulating knowledge acquisition through different strategies.

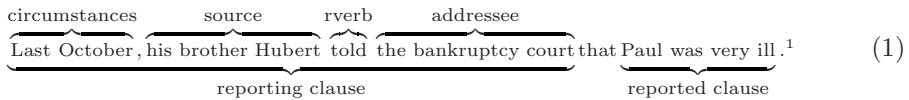
## 2.2 Extracting Reported Speech

The main source for our fuzzy believer stems from *reported speech* in newspaper articles. This allows us to explicitly attribute statements to sources. Additional information that can be analyzed and therefore has to be identified by the extracting component comprise the reporting verb and circumstantial information.



**Fig. 2.** Information after extracting reported speech – sources are isolated and topics (T) not yet identified

To find reported speech structures, we identified six patterns around 50 verbs [7] that are often used within reported speech constructs. Example 1 shows a typical reported speech structure and identifies the different elements.



This information can be utilized to perform evidential analysis [8], thereby assigning different degrees of *confidence* in a statement according to the reliability of the source and the reporting verb used.

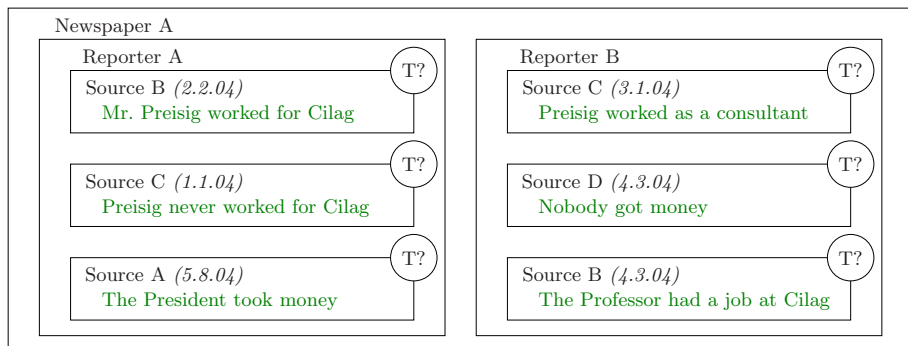
Figure 2 shows the results of the reported speech extraction component assuming 6 fictitious newspaper articles<sup>2</sup> dealing with two different topics. We adapted a presentation scheme for beliefs proposed by Ballim and Wilks [1], using nested boxes to visualize the held beliefs of different actors. Each box contains a statement together with its source and the publishing date. Every statement is assigned to the reporter who wrote the article containing the statement, and finally the newspaper who published the article is named.

## 2.3 Generating Profiles

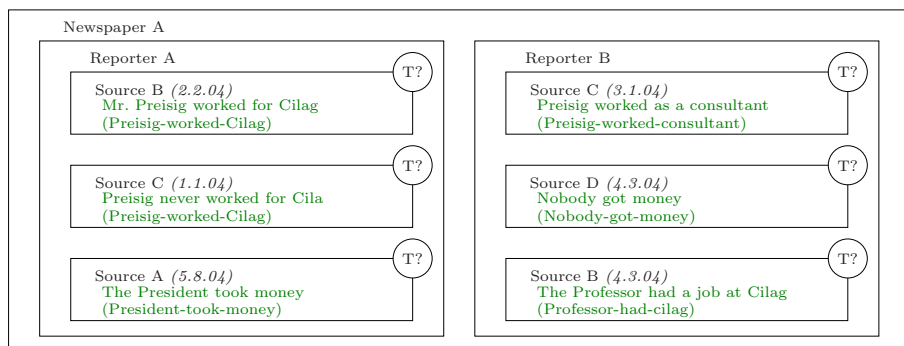
The profile generation component assembles the reported speech fragments and prepares them for the next processing step. A profile assigns each statement to a source, reporter, and newspaper. Basically, the component extracts the reported speech clauses, which can then be further processed by a parser. It also adds coreference information for each source by traversing the data structure created by our fuzzy coreference resolution system [5]. Figure 3 shows our example with the added coreference information.

<sup>1</sup> Sentence from Wall Street Journal 03.03.88.

<sup>2</sup> Inspired by articles in WSJ 12.03.86.



**Fig. 3.** The different statements after identifying the source entities



**Fig. 4.** The different extracted predicate-argument structures

## 2.4 Extracting Predicate-Argument Structures

To decide whether a sentence has the same topic as another one, we need to find a way to compare sentences with each other. To facilitate this task, we do not compare whole sentences, but their predicate-argument structures, consisting of “subject,” “verb,” and “object.” Because one sentence might contain more than one statement, a correct syntactic analysis is paramount for predicate-argument structure (PAS) generation. Our experiments showed that no single parser is consistently reliable enough for PAS extraction. Thus, our PAS extraction component can work with the results of three different parsers: RASP [9], MiniPar [10], and SUPPLE [11].

The PAS extractor applies a custom rule set for each of these parsers in order to determine subject, verb, and object of a statement.

The extracted predicate-argument structures for our example can be seen in Fig. 4. To demonstrate the system, we chose rather simple sentences containing only one PAS each but the algorithm can handle more complex structures as well.



## 2.5 Computing Beliefs

The core of our system is the *fuzzy believer* component. Its tasks are:

1. Identify a topic for each statement.
2. Compute the fuzzy representation for each statement to identify polarity.
3. Process fuzzy information for each topic according to a strategy.
4. Generate a graphical view of the result.

*Identifying Domains.* The first step is to group the statements into *domains* according to their topics. These domains constitute the basic sets for the fuzzy operations performed later on; basically, they partition the statement space into individual domains, which can be processed independently. Every domain represents one topic identified by the extracted PASs.

To determine if a statement fits into an existing domain, we use *heuristics* to measure the semantic proximity of each new statement with the statements in all existing domains. For this, the system applies two main heuristics: (1) A WordNet [12] related heuristic, and (2) a substring heuristic.

These heuristics compare the PAS elements of one statement with the elements of the other statements in one domain and return a value representing how similar the heuristics consider the two PAS elements. A runtime option defines if *strict* matching is necessary to include a new statement in a domain, or if a more *lenient* matching is sufficient. For a strict match, the new statement's PAS must be similar to all existing statements within a domain. In case of a lenient match, the new statement needs only to be similar to one statement of a domain, essentially implementing a transitive relation on the domain elements.

To cause a match between two statements, at least two parts of their corresponding PAS structures must be similar enough. That means, the value assigned by a heuristic must exceed the defined threshold for either subject and object, subject and verb, or verb and object.

This approach permits assigning a statement to more than one domain. If a new statement does not fit into any of the existing domains, a new domain is dynamically created, initially containing this statement.

Each domain contains all statements that have the same or opposite meaning. In other words, we try to identify each fact in the world and arrange all statements concerning this fact in one domain. The example in Fig. 5 should contain two domains after the classification process: *T1* "Someone taking money" and *T2* "Preisig working as consultant at Cilag." The different statements are assigned a label identifying the topic.

*Identifying Polarity.* In the next step, the statements gathered for each domain have to be evaluated by identifying their polarity. The goal is to identify opposing statements by using different fuzzy heuristics. The fuzzy representation  $\mu_{S_i}$  of a statement  $S_i$  contains the degrees of similarity of this statement with all other statements within the same domain. Each degree is normalized to a fuzzy value in the  $[0, 1]$ -interval and can be interpreted as the semantic distance between two statements. Fig. 6 shows the fuzzy representation of a statement  $S_1$  within a

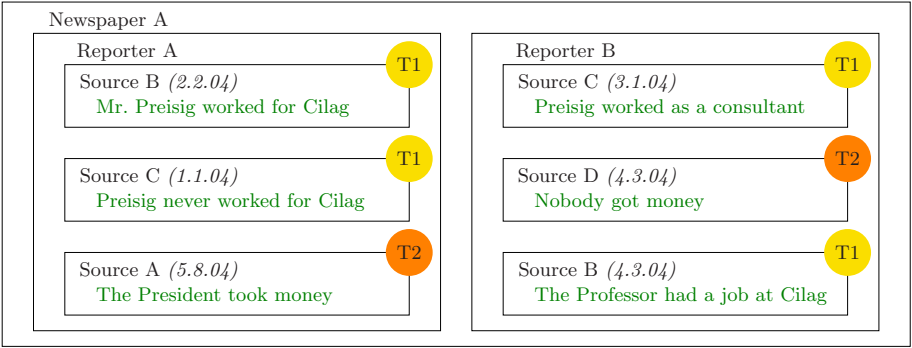


Fig. 5. The different topics after identifying the domains

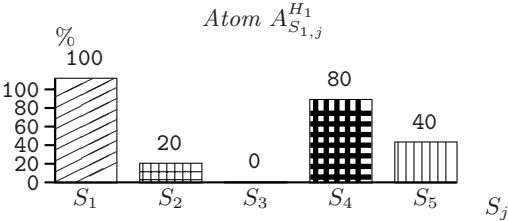
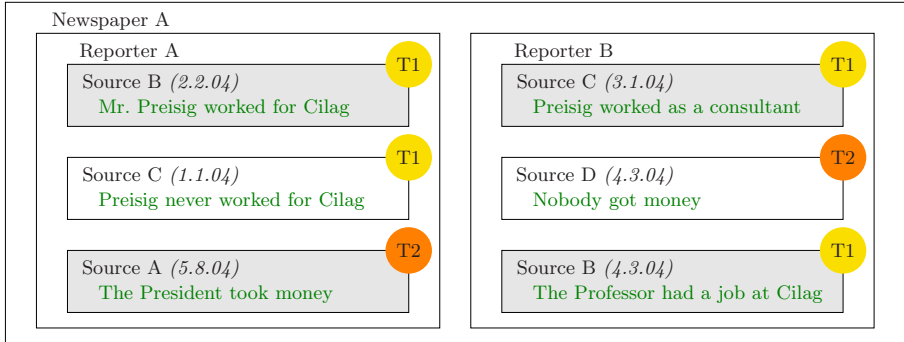


Fig. 6. Statement  $A_{S_{1,j}}$  with correlation grades for all statements in the domain ( $S_1, \dots, S_5$ ) as computed by heuristic  $H_1$

domain containing five statements ( $S_1, \dots, S_5$ ). The fuzzy sets are interpreted in a possibilistic fashion: A fuzzy value of 0 indicates no possible semantic similarity between the two statements, while a value of 1.0 indicates the highest possibility of similarity between them. In the current implementation, only one heuristic is used. It compares the verbs of two statements using their WordNet semantic distance to find synonyms and antonyms.

*Computing Beliefs.* One of the crucial parts of the fuzzy believer system is to decide which of the collected statements to believe and which to reject. For this, each domain is processed independently. The system implements the following strategies: (1) Believe majority, (2) believe old news, (3) believe new news, (4) believe certain source/reporter/newspaper, and (5) believe weighted majority. The strategies are based on fuzzy processing. Three fuzzy operations are essential to implement the strategies: *Merging*, *expanding*, and *revising*. These operations are computed directly on the fuzzy set representation of each statement, which has been generated as described above.

Based on the fuzzy representation, the merge operation groups all statements into one class, if a threshold of semantic similarity is reached. Usually, merging all statements leads to two classes within each domain, one containing statements about a topic and the other one containing opposing statements about this topic. The majority strategy picks the class with the most statements and marks them



**Fig. 7.** Believe Majority: The system believes statements with grey background

as belief. Fig. 7 shows the result of this strategy for our example. For topic T2 there is no majority, in this case the system chooses either of the statements.

The expansion operator initially believes the first statement in a domain and each new statement becomes included only if it is compatible with all the ones existing in a domain. Expansion [13] can be used to implement the “Believe old news” strategy by ordering the processing according to the publishing date.

For the fuzzy belief revision process [14], new statements are always believed and only those of the existing statements that are not in conflict with the new ones are kept. This is exactly what we need for the “Believe new news” strategy. For the weighted majority strategy, we use information of the majority strategy and combine it with information about the reliability of the newspaper, the reporter, and the source of the statement. To believe in a certain source, reporter, or newspaper, fuzzy processing is not necessary and this strategy can be implemented utilizing the profile generator information.

## 2.6 Summary

Our fuzzy believer system processes natural language articles and identifies the topics discussed in a text. Statements are extracted from the texts based on reported speech structures and assigned to domains, which form the formal basis for automatic processing using fuzzy operators. The main believer component can simulate different reading strategies, like a reader accepting all new information (and erasing conflicting old knowledge), or a stubborn reader clinging to old beliefs while rejecting all incompatible new information.

The output of the fuzzy believer system is a set of held beliefs and rejected beliefs acquired from “reading” a document collection. Presently, we export this result into a graphical representation using the LaTeX-format similar to the presentation of the examples in this paper.

**Table 1.** Domain Classification: Recall and Precision for different parsers

Configuration	Recall			Precision		
	Rasp	Minipar	Manual	Rasp	Minipar	Manual
3-3-3-lenient	0.59	0.54	0.56	0.57	0.63	0.78
3-3-3-strict	0.59	0.50	0.55	0.63	0.75	0.85
5-5-5-lenient	0.70	0.60	0.62	0.29	0.39	0.29
5-5-5-strict	0.52	0.52	0.52	0.41	0.53	0.54
5-3-5-lenient	0.65	0.51	0.59	0.31	0.57	0.45
5-3-5-strict	0.59	0.58	0.52	0.56	0.41	0.61

### 3 Evaluation

The system we present here is complex and attempts a novel analysis. Therefore no Gold standard corpora are available. In order to evaluate our system we have thus chosen to evaluate its components separately on standard reference resources in related domains.

*Extracting Reported Speech.* In order to evaluate the reported speech extraction component, we randomly picked 7 newspaper articles from the Wall Street Journal corpus. The articles contain about 400 sentences ( $\sim 6100$  words), among them 133 reported speech constructs. For the detection of reporting verb and source, our system achieved a recall value of 0.83 and a precision value of 0.98. This results in an f-measure of 0.90.

*Identifying Domains.* The domain finding task is quite hard and error-prone. Remember that the domain classification is solely based on the predicate-argument structures extracted from the output of one of the three deployed parsers. The evaluation of the domain finding component includes a comparison of the results obtained with RASP, MiniPar, and manually annotated predicate-argument structures (gold standard). The conservative strategy of SUPPLE, which only marks relations that are considered to be 100% correct, proved to be not applicable, as it creates too few extractable PAS.

The test data we used is taken from the MSR corpus [15] and comprised 300 paraphrase pairs. We assumed all sentences were reported clauses to skip the reported speech extraction part from distorting the domain finding evaluation results. The special layout of the test corpus, containing pairs of paraphrases and thus two statements per topic, made it necessary to develop a method to measure the performance accurately. The fact that one sentence can contain more than one statement, represented as different predicate-argument structures, made the evaluation scenario more complex. We conducted a test with a test set of 116 paraphrase pairs, which was additionally annotated by hand with predicate-argument structures. This allows an estimation on the influence of the parser

and the parser extraction component on the domain classification process. The results can be found in Table 1.<sup>3</sup>

*Identifying Polarity.* To test the sense detection or opinion grouping function, we would need a special corpus containing test data with opposing and supporting statements for a special opinion, which are semantically close enough to fulfill the requirement of belonging to the same domain. The data that comes closest to these conditions are the entailment pairs of the PASCAL challenge corpus [16]. There are some minor drawbacks, though.

Firstly, the positive entailment examples are rather easy to evaluate, because if one sentence entails another, the senses of the two sentences must have the same direction. But non-entailment between two sentences doesn't necessarily imply opposing opinions in these sentences. But fortunately this is often the case for the PASCAL-2 challenge corpus we used. A second problem is the fact that sentence pairs, especially without entailment, would not be assigned the same domain by our domain classification algorithm, therefore it is not possible to evaluate the data using only the polarity identification component.

We solved these problems by checking the non-entailing examples manually for opposing sentences and developing a scheme to measure the performance of the sense detection algorithm without influence from the domain finding component. This scheme comprises the consideration of only those statement pairs that are correctly assigned to the same domain.

We tested different configurations and computed accuracy for two different settings. For one experiment, we included all results in the evaluation, counting the entailment pairs that were not grouped into the same domain by the domain classification as non-entailing. In the table, this is referred to as "Sense & Domain." The other test setting only considered the sentence pairs that were actually grouped into the same domain by the domain classification component. That way, we limited the influence of the domain classification algorithm on the sense detection. An overview of the achieved performance is shown in Table 2<sup>3</sup> with the additional configuration parameter showing the threshold for assigning the same polarity to a statement.

## 4 Related Work and Discussion

The extraction of opinions from newspaper articles [17] or customers reviews [18, 19] has become an active research field. Those approaches are usually only concerned with the identification and extraction of information without processing it further, except for binary classification within a clearly specified domain.

In the wake of the PASCAL challenge [20, 16], systems have been developed to deal with the relation of sentences to each other. The different approaches include

---

<sup>3</sup> The configuration settings in the table mean, from left to right: Maximum WordNet Distance between (1) subjects, (2) verbs, (3) objects of two statements. And (4) indicates whether a new statement has to match with one (lenient) or all (strict) statements within one domain.

**Table 2.** Polarity Identification: Accuracy values for different parse methods

Configuration	Accuracy			
	Sense & Domain		Only Sense	
	Rasp	Minipar	Rasp	Minipar
3-3-3-strict-0.7	0.52	0.55	0.53	0.58
5-5-5-lenient-0.7	0.51	0.53	0.51	0.53
5-5-5-strict-0.3	0.52	0.53	0.55	0.51
5-5-5-strict-0.7	0.51	0.54	0.50	0.56
7-7-7-strict-0.7	0.51	0.52	0.51	0.52

the recognition of false entailment [21], or learning entailment [22]. Others are concerned with relatedness between words and how to measure it [23]. We were not interested in concentrating on one of these areas but rather to develop an all-embracing system incorporating different aspects.

The results our system achieved for extracting reported speech is highly competitive. Doandes [24], using a different subset of the WSJ-corpus, reports a recall of 0.44 and a precision of 0.92 for their system compared to 0.83 and 0.98 our system obtained.

For the domain classification, our best results for 300 paraphrase pairs from the MSR-corpus are: Precision 38%, Recall 81% and Precision 52%, Recall 58%. These values can probably be improved by using more sophisticated heuristics, although there will be a ceiling set by the parser and by the use of language in general. The same meaning can be expressed by various different sentences whose words are not in close relations to each other and therefore hard to detect by current NLP tools. Keeping these facts in mind, the obtained numbers are rather satisfactory and promising for future development.

The rather shallow semantic approach sets a practical limit to the achievable results. This can be inferred by comparing the numbers obtained using manually parsed predicate-argument structures with the numbers obtained by the parsers. It shows that there is space for improvement on the side of the parsers, as well as on the side of the PAS extractor. Combining the results of different parsers could also lead to better results, but a precision of 55% and a recall of 85%, as obtained for the best configuration of the system using manually parsed PASEs, shows that it needs more and/or better heuristics to get a really significant improvement.

The polarity identification task was expectedly the hardest one. This is illustrated by the rather poor results we obtained by trying to find different opinions within one domain. Best accuracy values were obtained using Minipar and were around 58%. This task is very hard for computational systems. But with more elaborated heuristics it is possible to increase these numbers, comparable to the Pascal challenge [20, 16], where systems also started with around 50% accuracy and improved over time.

Testing of the different strategies revealed that the fuzzy processing operators perform in accordance to their assigned tasks. Further evaluation of the results would need some kind of measure to get quantitative, comparable results. This is beyond the scope of this paper and deferred to future work.

## 5 Summary and Conclusions

We presented a fuzzy believer system, which is capable of differentiating between different topics and different polarity of statements and decides what to believe based on configurable strategies. The system was applied to processing reported speech information, generating a belief set containing the knowledge obtained from “reading” different newspaper articles. Our approach is based on the application of fuzzy set theory to natural language processing resulting in a fuzzy believer with variable belief strategies.

The results for the individual subtasks are promising but the development of a measure to evaluate the system as a whole is still pending. The growing number of available news sources, blogs, and webpages makes it necessary to facilitate the information gathering for humans. Our fuzzy believer was designed to deal with huge amounts of information and supports a user’s opinion finding process.

## References

1. Ballim, A., Wilks, Y.: *Artificial Believers: The Ascription of Belief*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA (1991)
2. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A framework and graphical development environment for robust NLP tools and applications. In: *Proc. of the 40th Anniversary Meeting of the ACL*. (2002)
3. Bergler, S.: Conveying attitude with reported speech. In Shanahan, J.C., Qu, Y., Wiebe, J., eds.: *Computing Attitude and Affect in Text: Theory and Applications*. Springer Verlag (2005)
4. Bergler, S., Doandes, M., Gerard, C., Witte, R.: Attributions. [25] 16–19
5. Witte, R., Bergler, S.: Fuzzy Coreference Resolution for Summarization. In: *Proc. of 2003 Intl. Symposium on Reference Resolution and Its Applications to Question Answering and Summarization (ARQAS)*, Venice, Italy (June 23–24 2003) 43–50
6. Gerard, C.: *Modelling Readers Of News Articles Using Nested Beliefs*. Master’s thesis, Concordia University, Montréal, Québec, Canada (2000)
7. Quirk, R.: *A comprehensive grammar of the English language*. Longman Group Limited (1985)
8. Bergler, S.: *The Evidential Analysis of Reported Speech*. PhD thesis, Brandeis University, Massachusetts, USA (1992)
9. Briscoe, E., Carroll, J., Watson, R.: The Second Release of the RASP System. In: *Proc. of the COLING/ACL 2006 Interactive Presentation Sessions*. (2006)
10. Lin, D.: Dependency Based Evaluation of MINIPAR. In: *Proc. of the Workshop on the Evaluation of Parsing Systems, First Intl. Conference on Language Resources and Evaluation*. (1998)
11. Gaizauskas, R., Hepple, M., Saggon, H., Greenwood, M.A., Humphreys, K.: SUPPLE: A practical parser for natural language engineering applications. In: *Proc. of the 9th Intl. Workshop on Parsing Technologies (IWPT2005)*, Vancouver (2005)
12. Fellbaum, C., ed.: *WordNet: An Electronic Lexical Database*. MIT Press (1998)
13. Witte, R.: *Architektur von Fuzzy-Informationssystemen*. BoD (2002) ISBN 3-8311-4149-5.
14. Witte, R.: Fuzzy Belief Revision. In: *9th Intl. Workshop on Non-Monotonic Reasoning (NMR’02)*, Toulouse, France (April 19–21 2002) 311–320

15. Dolan, B., Brockett, C., Quirk, C.: Mirosoft research paraphrase corpus. Online: [http://research.microsoft.com/research/nlp/msr\\_paraphrase.htm](http://research.microsoft.com/research/nlp/msr_paraphrase.htm) (March 2005)
16. Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., Szpektor, I.: The Second PASCAL Recognising Textual Entailment Challenge. In: Proc. of the Second PASCAL Challenges Workshop on Recognising Textual Entailment. (2006)
17. Bethard, S., Yu, H., Thornton, A., Hatzivassiloglou, V., Jurafsky, D.: Automatic extraction of opinion propositions and their holders. [25] 20–27
18. Gamon, M., Aue, A., Corston-Oliver, S., Ringger, E.K.: Pulse: Mining customer opinions from free text. In Famili, A.F., Kok, J.N., Peña, J.M., Siebes, A., Feelders, A.J., eds.: Advances in Intelligent Data Analysis VI, 6th International Symposium on Intelligent Data Analysis, IDA 2005, Madrid, Spain, September 8–10, 2005, Proc. Volume 3646 of LNCS., Springer (2005) 121–132
19. Kim, S.M., Hovy, E.: Identifying and analyzing judgment opinions. [26] 200–207
20. Dagan, I., Glickman, O., Magnini, B.: The PASCAL Recognising Textual Entailment Challenge. In: Proc. of the PASCAL Challenges Workshop on Recognising Textual Entailment. (2005)
21. Snow, R., Vanderwende, L., Menezes, A.: Effectively using syntax for recognizing false entailment. [26] 33–40
22. MacCartney, B., Grenager, T., de Marneffe, M.C., Cer, D., Manning, C.D.: Learning to recognize features of valid textual entailments. [26] 41–48
23. Klebanov, B.B.: Measuring Semantic Relatedness Using People and WordNet. [26]
24. Doandes, M.: Profiling For Belief Acquisition From Reported Speech. Master's thesis, Concordia University, Montréal, Québec, Canada (2003)
25. Qu, Y., Shanahan, J., Wiebe, J., eds.: Exploring Attitude and Affect in Text: Theories and Applications. Technical Report SS-04-07, Stanford, California, USA, AAAI Press (March 22–25 2004)
26. Proc. of the Human Language Technology Conference of the NAACL, New York City, USA, Association for Computational Linguistics (June 2006)



# Rethinking the Semantics of Complex Nominals

Nabil Abdullah and Richard A. Frost

School of Computer Science, University of Windsor, Windsor,  
Ontario N9B 3P4, Canada  
{nabdullah,richard}@cs.uwindsor.ca

**Abstract.** Complex Nominals (CNs) have simple syntactic structure that conceals non-trivial semantic characteristics. While speakers of natural languages combine noun(s)/adjective(s) with a head noun to indicate existing or novel concepts with ease, formalizing such a semantic process, however, has proven to be a daunting task. In this paper, we present a unified semantic approach for constructions involving a head-noun and modifier(s), i.e., adjective(s)/noun(s). Based on a rigorous typing system, this approach uses set intersection as the only underlying semantic rule. We argue that this novel approach is compositional and warrants consistent inferences.

## 1 Introduction

A CN is a sequence of one or more nouns or adjectives preceding a head noun.<sup>1</sup> Instances of such a construction include *apple pie*, *information retrieval system*, *computer book sale*, *former political activist*, etc. CNs have been investigated in many fields: linguistics (e.g., [3, 12, 19, 16, 5, 17]), cognitive science (e.g., [4, 8, 18]), and AI ([2, 11, 7]), among others. The bulk of such research in AI and linguistics is concerned with identification and classification of the semantic relations in noun-noun constructions. Other problems such as adjective-noun combinations and bracketing are treated separately. Each field, understandably, uses different tools to address the respective concerns. Corpus analysis is by far AI's most commonly used tool in the treatment of CNs.

Although corpus-based systems have met some successes, a more systematic approach to CNs is yet to emerge. Such a semantic approach must address two fundamental questions: what are the semantic values of common nouns, and adjectives? And what are the semantic values of their combinations?

What makes the semantic analysis of CNs difficult is that some modifiers are reference-modifying (i.e., intensional), some are referent-modifying (i.e., extensional), and some can be both. The difficulty of analysis increases when modification involves multiple modifiers.

We address these issues and present a formal system that, we argue, is capable of overcoming limitations posed to corpus-based approaches. In addition, as AI workers, we take the issue of machine-implementability seriously into consideration, without compromising the philosophical and linguistic concerns.

---

<sup>1</sup> This use of the term “complex nominal” is more general than is commonly used in the literature, e.g., in [17] modifiers are limited to nouns and non-predicating adjectives.

## 2 On Compositionality

Also known as *Frege's Principle*, *compositionality* is informally stated<sup>2</sup> as: the meaning of a whole expression is a function of the meaning of its constituents and their syntactic mode of combination. In formal languages this principle is taken for granted. In computer programming, it is implicitly assumed when proving some aspects of programs such as correctness and termination. Within natural languages, however, it is debatable. While the bulk of language expressions seem to adhere to compositionality, it is obvious that some expressions such as idioms and lexicalized expressions, e.g., *white wine*, *potential energy*, etc. deviate from it. However, generally speaking, idioms and lexicalized expressions are very limited in number within a language and are normally listed as entries in a conventional dictionary. Thus, they are treated as lexemes and consequently pose no threat to compositionality. However, as [13] points out, what remains controversial is where to draw the line between compositional expressions, on one side, and lexicalized/idiomatic expressions, on the other side.

In Richard Montague's highly formalized semantic theory, compositionality occupies centre stage. It is expressed as rule-to-rule correspondence between the syntax and semantics. That is, for every rule of the syntax, there is a corresponding semantic rule for computing its meaning. For example, given the syntactic rule (1), in BNF notation, a possible corresponding semantic rule is (2):

- 1)  $CN ::= M N$ , where  $M$  stands for a modifier      2)  $\| CN \| = F (\| M \|, \| N \|)$

That is, given a CN, its meaning is determined by the meanings of the modifier and noun involved, whatever they might be, plus the way they are linearly organized, namely a head noun preceded by a modifier.

Our views regarding compositionality parallel that of Frege's and Montague's. For a non-compositional approach would face numerous problems, see [10] for examples of non-compositional semantic approaches. In section (6) we show how to account for compositionality by taking the semantic values of the parts to be typed sets, and the function  $F$  a set intersection.

## 3 On the Semantic Theory

Lexical categories such as nouns and adjectives belong to the set of linguistic realm. They acquire meanings when they correlate to entities in the world or model. Thus, the semantics of a language can be thought of as a process by which a link between the linguistic and the extra-linguistic is established. This is the approach taken by extensional semantic theories.

In a purely-extensional semantic theory, the meaning of a term in a language is taken to be the denotation of that term within a world or a model. Based on such a theory, co-extensive terms can be (counter-intuitively) interpreted to have the same meaning. For example, the terms *the largest integer*, *dragon* and *unicorn* are interpreted to mean the same thing.

---

<sup>2</sup> See [13, p 135] on suggestions as to how to make this definition precise.

Intensional semantics overcomes this difficulty by taking meaning to be composed of intensions as well as extensions. Intensions are considered as functions with indices as domains. One of the indices can be *possible worlds*. It is then said that intensions determine extensions. With the inclusion of intensions, a semantic theory will not admit the terms *the largest integer* and *unicorn* as having the same meaning. For in some other world the set of unicorns might not be empty.

Thus, a semantic framework that invokes notions such as possible worlds seems to be ideal for CNs, e.g., Montague took the meanings of adjectives to be functions from intension to extension, see [13] and [22]. Yet, as [10] points out NLU systems based on Montague semantics with its use of infinite sets and functions to functions render developing such systems impossible.

We will show that the formalism we are proposing in this paper is capable of capturing intensions, in some sense, and yet is still machine-implementable.

## 4 On the Semantic Values of Nouns and Adjectives

In the previous section, we pointed out that our goal is a compositional account of CNs. We hinted that this can be achieved via the function  $F$ . However, we have not provided any characterization of its arguments. That is, the values of the objects  $\|M\|$  and  $\|N\|$ . This brings up the question regarding the nature of the denotations of adjectives and nouns. Jespersen [12] (*see also* [21]) provides the following account regarding adjectives and nouns (termed ‘substantives’ by Jespersen):

*On the whole, substantives are more special than adjectives, they are applicable to fewer objects than adjectives, in the parlance of logicians, the extension of a substantive is less, and its intension is greater than that of an adjective.*

The difference between adjectives and that of common nouns has also been expressed by Strawson (*see*, [9]) as that between *sortal* and *nonsortal* predicates. Strawson states that a sortal predicate, “supplies a principle for distinguishing and counting individual particulars which it collects”, while a nonsortal predicate “supplies such a principle only for particulars already distinguished, or distinguishable, in accordance with some antecedent principle or method”.

Jespersen’s and Strawson’s accounts shed some light into the possible semantic values for both nouns and adjectives. Jespersen’s suggests, among other things, that adjectives have wider “applicability” and Strawson suggests, among other things, that adjectives denote properties. Both analyses seem to be in accord and are plausible in adjective-noun constructions. The formalism we propose next takes these observations into consideration.

## 5 On Modification

Despite the ontological differences between nouns and adjectives, they have a common functionality of interest: modification of head nouns. This functionality is evident at both the syntactic and semantic levels. In the case of the former, both nouns and adjectives can occupy the attributive position, i.e., preceding a head noun. At the

semantic level, although a noun and an adjective modifier may contribute differently to the meaning of the compound, a near-universal truism is that the denotation of the compound is a subset of the denotation of its constituent head noun. Expressions admitting such reading are called *subsectives*. In some special cases, the denotation of the compound is a subset of both the denotation of the head noun and the denotation of the modifier. This is called the *intersective* reading. Modifiers that allow intersective reading are, also, referred to as referent-modifying (i.e., extensional), while those that do not allow such a reading are called reference-modifying, i.e., intensional, see [22] for details. Table 1, where ‘M’ and ‘H’ stand, respectively, for modifier and head noun, illustrates by example the points raised in this paragraph.

**Table 1.** Similarity of modification in nouns and adjectives

Noun-Noun example	$\ M\ H\ $ $\subseteq \ H\ $	$\ M\ H\ $ $\subseteq \ M\ $	Modifier: int/ext	Adjective- Noun example
player coach, child murderer <sup>3</sup>	Y	Y	Ext	Canadian coach
soccer game	Y	N	Int	competitive game
Glass cup	Y	Y	Ext	red car
elephant chocolate	Y	N	int	toy store <sup>4</sup>

We believe that such a common behavior warrants a uniform semantic treatment that takes into account the intensional and extensional behavior of modifiers. This, as compositionality demands, should be expressed by providing a single semantic rule that mirrors the syntactic one. In our system, such a rule is set intersection.

## 6 The Formalism

In this section we present a formal system for CNs. It starts by introducing typed-sets: the semantic values for the parts in a complex expression. Next, it defines rules, axioms, and the typing system for handling complex compounds. In devising the formalism, we have taken the following factors into considerations:

- The duality of the semantic behavior of nouns: a noun can be in the head or the modifier positions of a CN, e.g., *boat house* and *house boat*.
- The bracketing problem.
- The general applicability of adjectives.
- The desire to strike a balance between simplicity and machine- implementability of extensional semantics and the sophistication of intensional ones.

The resulting system, we argue, has the following characteristics:

- 1) Captures meanings’ two aspects, i.e., intension and extension.
- 2) Is machine-implementable.
- 3) Can fit within logical frameworks such as that of Montague’s.

<sup>3</sup> This expression can mean “one who murders children” or “a child who commits murder”. It is the latter reading that is considered here.

<sup>4</sup> For discussion of subsective reading of privatives, see [1] and for linguistic evidence see [20].

- 4) Is Russell-Paradox-free.
- 5) Is an extension to and/or modification of set theory or lambda calculus notations, which are well studied and used.

### 6.1 The Case for Typed-Sets

The Cantorian notion of a set has been affectionately embraced by mathematicians: "No one shall expel us from the paradise which Cantor created for us", as David Hilbert succinctly puts it. While it is hard to disagree with Hilbert wholesale, one can argue that a set, as conceived and defined by Cantor, makes rather simplistic assumptions. One can identify at least two shortcomings: an element is either in the set or not, that is, a world with sharp boundaries; an element once in a set is completely identified with it. That is, the only property that a member has is its membership in the respective set, all other properties are "lost"—individuality is sacrificed for plurality. The former, was addressed by fuzzy sets. The latter we address here by adding types for both members of the set and the set itself. This newly conceived set is termed a "typed-set". We argue that one application of such an abstract typed-set is to model modification in CNs. We argue that this modification allows an extensional representation of terms usually described as intensional while at the same time preserving the original Cantorian conception of the set. We will show that this augmentation of the classical set allows representations of terms that are "normally" hard to represent such as frequency terms, e.g., *occasional*, and non-committal terms, e.g., *alleged*. Such a representation is not possible using the classical set or the fuzzy one. This is accomplished while maintaining the original classical set properties, i.e., typed sets can be seen as a generalization of classical sets.

**(Typed Sets)** A typed set is a set-theoretic set that has the following form:

$$M = \{e : \alpha, \dots\} : \beta \quad (1)$$

Here,  $M$  is a set of type  $\beta$ . Its member  $e$  is of type  $\alpha$ . It should be noted here that the set member and its type are one integral whole.<sup>5</sup> In lambda notation, this can be expressed as in (2), where  $\Gamma$  is a context or environment:

$$\begin{array}{l} \Gamma, e : \alpha \vdash e : \alpha \qquad \Gamma, M : \beta \vdash M : \beta \\ (\lambda x. M(x)) (e : \alpha) \end{array} \quad (2)$$

From (1) and (2), it should be obvious that set members are first-order elements, i.e. individuals. Thus, a set cannot be a member of another set of the same order, including itself.

### 6.2 The Semantic Values of Complex Expressions

It is assumed that the types  $\alpha$  and  $\beta$  in (1) range over types in a type hierarchy, where each node in the hierarchy represents a type. A fully-fledged system will need such a hierarchy. However, for our illustrative purpose we will be using the following subset of types:

---

<sup>5</sup> One may think of this as an approximation of the Aristotelian view of a *universal* inhering in a *particular*.

- The type “ $\top$ ”, assigned to extensional<sup>6</sup> terms, e.g., *red*, *Canadian*, etc.
- The type “ $\perp$ ”, the absurd type, is assigned to “non-committal” terms, e.g., *potential*, *alleged*, *presumed*, etc.
- The type “role”, is assigned to common nouns denoting roles, e.g., *senator*, *employee*, *manager*, *dancer*, etc.

Given a type  $\top$  that is assigned to extensional terms, then, we can define the classical set to be of that type.

**(Classical Set)** A classical set is a typed set which is of type  $\top$  and all its members, if any, are of type  $\top$ .

**(Subtype Relation  $\ll$ )** Types in the hierarchy are constrained by the partial-order subtype relation indicated by  $\ll$ . The expression  $\alpha \ll \beta$  is read “ $\alpha$  is a subtype of  $\beta$ .”

For example the subtyping relation that holds between the types *employee* and *manager* and *role* can be expressed as follows:

$$\text{manager} \ll \text{employee} \ll \text{role}$$

**(Complex Type)** The type hierarchy contains the simple types. To handle cases where more than one modifier is involved in a CN, we need a systematic way to assign types that mirrors the complexity of the expressions.

**Definition (Types).** Assume a (non-empty) set of simple types  $\tau$ . Define the set  $T$  of types as follows:

- i)  $\tau \subset T$
- ii) if  $\alpha$  and  $\beta \in T$  then  $(\alpha:(\beta)) \in T$

According to these type-forming rules, the expression *deep blue sea*, will be assigned the types  $(\text{deep}:(\text{blue}:(\text{sea})))$  and  $((\text{deep}:(\text{blue})):(\text{sea}))$  or by removing redundant parentheses  $\text{deep}:(\text{blue}:\text{sea})$  and  $(\text{deep}:\text{blue}):\text{sea}$ . These complex types correspond, respectively, to the wide- and narrow-scope readings of the adjective *deep*.

### 6.3 Operations on Typed Sets

In this subsection, we define the basic set operations and related axioms necessary to compute the meaning of CNs. This includes set intersection, equality of sets, and equality of particulars (or set members).

**(Convention)** Greek letters stand for types, lower-case letter for members, and upper-case letters for sets.

**(The most specialized types)**  $\perp \ll \alpha$  ( $\perp$  is the lower-most type)

**(Type of complex expressions)** The type of a complex expression is the right-most:

Given set  $X$ :  $\alpha$ :

---

<sup>6</sup> For classification of adjectives, see [1] and [13].

$$\frac{X:\alpha}{X: \mathfrak{I}(\alpha)}$$

Where  $\mathfrak{I}$  is a function defined as follows:

$$\mathfrak{I}(\alpha: (\beta)) = \begin{cases} \beta, & \text{if } \beta \in \tau \\ \mathfrak{I}(\beta), & \text{otherwise} \end{cases} \quad \text{where } \tau \text{ is the set of simple types.}$$

**(Particulars)** Since a particular (or a set member) and its type are an integral whole, we define equality between them using the symbol ' $\equiv$ '. Given particulars  $e_1:\alpha$  and  $e_2:\beta$

$$e_1 \equiv e_2 \iff e_1 = e_2 \wedge [(\alpha = \beta) \vee (\alpha \ll \beta)] \quad (\text{particular equality})$$

Using this rule, we can say that a cat is an animal, but not the other way around.

**(Subset)**  $X:\alpha \subseteq Y:\beta \iff$

$$(\forall e: \sigma) e: \sigma \in X: \alpha \Rightarrow [(e: \sigma \in Y: \beta) \vee (\exists \rho) \rho \in T \wedge e: \rho \in Y: \beta \wedge (\sigma \ll \rho)]$$

**(Set equality)**  $X:\alpha = Y:\beta \iff [(X:\alpha \subseteq Y:\beta \wedge Y:\beta \subseteq X:\alpha) \wedge (\alpha = \beta)]$

$$\frac{(e_1: \top) \in X: \alpha \quad (e_2: \alpha) \in Y: \beta \quad e_1 = e_2}{Z = X \cap Y \text{ s.t. } (e_1: \alpha) \in Z: \alpha: (\beta)} \quad \textbf{(Extensional intersection)}$$

Where *s.t.* stands for such that. This rule is used with extensional modifiers. Using this rule, we can validly deduce that *Mary is a Canadian dancer* from *Mary is beautiful dancer* and *Mary is Canadian*. For the adjective *Canadian* is extensional.

$$\frac{(e_1: \rho) \in X: \alpha \quad (e_2: \sigma) \in Y: \beta \quad e_1 \equiv e_2}{Z = X \cap Y \text{ s.t. } (e_1: \sigma) \in Z: \alpha: (\beta)} \quad \textbf{(non-extensional intersection)}$$

This rule blocks wrong inferences due to reference-modifying modifiers. Examples of this rule are in the next section.

$$\frac{X: \alpha \quad Y: \beta \quad Z = X \cap Y = \emptyset}{Z: \alpha: (\beta)} \quad \textbf{(\emptyset-Intersection)}$$

This rule allows us to speak about predicates with empty extensions, e.g., *white unicorn*. An empty set,  $Z$ , can be formed with the appropriate type.

## 6.4 Computing the Meaning of Compounds

Now that typed-sets and operations on them have been defined, in this section we will demonstrate how to compositionally compute the meaning of CNs. Since CNs are instances of one syntactic rule that states that a head noun can be preceded by some modifier(s), we argue, as compositionality demands, that a single semantic rule,  $F$ , corresponding to a single syntactic rule can be defined in terms of typed-sets and set intersection. For the simplest case where a head noun is modified by a single modifier,  $F$  can be defined as follows:

$F(MN) = \|M\| \cap \|N\|$ , where  $\|M\|$  and  $\|N\|$  are typed-set-valued functions

Since the modifier ‘M’ may involve more than one adjective or noun, the definition of  $F$  is refined to the following:<sup>7</sup>

$$F(MN) = \begin{cases} \|N\|, & \text{if } (M = \text{null}) \\ \|M\| \cap F(N), & \text{if } (\|M\| = X : \alpha \wedge (\alpha \in T)) \\ F(M) \cap \|N\|, & \text{if } (\|N\| = X : \beta \wedge (\beta \in T)) \\ F(M) \cap F(N), & \text{otherwise} \end{cases}$$

where  $T$  is the set of types

The domain of  $F$  is a bracketed expression. Such an expression can be obtained from a parse tree. For instance, the expression *liberal party scholarship scandal* can have the parse tree depicted in Fig. 1. This corresponds to the reading  $[[[ \text{liberal party} ] \text{scholarship} ] \text{scandal}]$ .

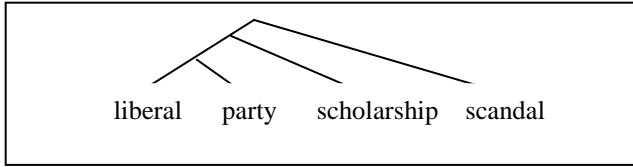


Fig. 1. Tree representation of *liberal party scholarship scandal*

Eliminating the outermost brackets, such an input can have one of the three forms:  $M[N]$ , i.e., ‘M’ is single word and ‘N’ multiple-words,  $[M][N]$ , i.e., both ‘M’ and ‘N’ contains multiple words, and  $[M]N$ , i.e., ‘M’ contains multiple words and ‘N’ is a single word.

To demonstrate how compounds can be represented and computed, consider the utterances *Fido is a dog*, *Fido is clever*, *John is a man*, and *John is clever*. The adjective *clever* and the nouns *dog* and *man* can be represented, given some state of affairs where John is the only man and Fido is the only dog, as follows:

$$\begin{aligned} M &= \{j:\text{man}\}:\text{man} & D &= \{f:\text{dog}\}:\text{dog} \\ C &= \{j:\text{man}, f:\text{dog}\}:\text{clever} \end{aligned}$$

Where:  $M$ ,  $D$ , and  $C$  denote, respectively, the sets of men, dogs, and clever things  
 $\| \text{John} \| = j$ , and  $\| \text{Fido} \| = f$

It should be noted how Jespersen’s definition is captured in the case of the adjective *clever*. The set  $C$  contains in its extension things that are said to have something in common: John is clever as a man, while Fido is clever as a dog. What they have in common is “cleverness”.

Given the data regarding Fido and John, we can provide truth-conditions for the utterance *Fido is a clever dog*. This statement is true if, and only if, the following boolean expression is true:

$$\begin{aligned} (f:\text{dog}) &\in \| \text{clever dog} \| \\ &= F(\text{clever}[\text{dog}]) = \| \text{clever} \| \cap F(\text{dog}) = \| \text{clever} \| \cap \| \text{dog} \| \end{aligned}$$

<sup>7</sup> In fact  $F$  can be abbreviated to

$$F(MN) = \begin{cases} \|N\|, & \text{if } (M = \text{null}) \\ F(M) \cap F(N), & \text{otherwise} \end{cases}.$$



$$\begin{aligned}
&= \{j: \text{man}, f: \text{dog}\} : \text{clever} \cap \{f: \text{dog}\} : \text{dog} \\
&= \{f: \text{dog}\} : \text{clever} : (\text{dog}) \text{ (by the non-extensional intersection rule)} \\
&\quad \text{True}
\end{aligned}$$

Now, assume that *Fido is a veteran shepherd*. This can be represented as follows:

$$\| \text{shepherd} \| = \{f: \text{shepherd}\} : \text{shepherd} \quad \| \text{veteran} \| = \{f: \text{shepherd}\} : \text{veteran}$$

It should be noted that a wrong inference such as *Fido is a clever shepherd* cannot be deduced, as desired. To further demonstrate our approach, consider the following utterances *John is a bank manger* and *John is an excellent employee*. A representation of these utterances results in the sets:

$$\begin{aligned}
\| \text{bank} \| &= \{j: \text{manager}, \dots\} : \text{bank} \\
\| \text{manager} \| &= \{j: \text{manager}\} : \text{manager} \\
\| \text{excellent} \| &= \{j: \text{employee}, \dots\} : \text{excellent} \\
\| \text{employee} \| &= \{j: \text{employee}\} : \text{employee}
\end{aligned}$$

Next, we try to get answers to the following questions: Is John an excellent manger? Is John a bank employee? These can be translated, respectively, into the following forms, numbered (1) and (2):

$$\begin{aligned}
(j: \text{manager}) \in \| \text{excellent manager} \| &= F(\text{excellent} [\text{manager}]) \\
&= \| \text{excellent} \| \cap F(\text{manager}) = \| \text{excellent} \| \cap \| \text{manager} \| \\
&= \{j: \text{employee}, \dots\} : \text{excellent} \cap \{j: \text{manager}\} : \text{manager} \\
&= \{ \} : \text{excellent} : \text{manager} \text{ (by } \emptyset \text{-Intersection)} \\
&\quad \text{False}
\end{aligned} \tag{1}$$

$$\begin{aligned}
(j: \text{employee}) \in \| \text{bank employee} \| &= F(\text{bank} [\text{employee}]) \\
&= \| \text{bank} \| \cap F(\text{employee}) = \| \text{bank} \| \cap \| \text{employee} \| \\
&= \{j: \text{manager}\} : \text{bank} \cap \{j: \text{employee}\} : \text{employee} \\
&= \{j: \text{employee}\} : \text{bank} : \text{employee} \\
&\text{(by non-extensional intersection, by } \equiv, \text{ and because manager } \ll \text{employee)} \\
&\quad \text{True}
\end{aligned} \tag{2}$$

In these examples, it should be clear how the inference and typing rules are working in tandem in the semantic evaluation process.

## 6.5 Capturing Non-committal Set Membership

One advantage of using typed-sets is the representation of entities that are associated with some predicate but which do not fall within its extension. In other words, referents of such a predicate have a “pending” membership in the set denoted by the predicate. Terms of modal import possess such a phenomenon: e.g., *potential*, *possible*, *presumed*, etc. fall into this category. To illustrate, consider the utterances *Hillary Clinton is a popular senator and a potential president* and *George Bush is a popular president*. These utterances can be represented as follows:

$$\begin{aligned}
\| \text{senator} \| &= \{h: \text{senator}, \dots\} : \text{senator} \\
\| \text{potential} \| &= \{h: \perp\} : \text{potential}
\end{aligned}$$

$$\begin{aligned} ||\text{president}|| &= \{ h: \perp, g: \text{president} \} : \text{president} \\ ||\text{popular}|| &= \{ g: \text{president}, h: \text{senator} \} : \text{popular} \end{aligned}$$

where

$||\text{Hillary Clinton}|| = h$ ,  $||\text{George Bush}|| = g$ , and “ $\perp$ ” is the subtype of every type

Looking at the set representation of the president predicate, we notice that the individuals  $h$  and  $g$  have different associations with the set. The latter has a full membership, for its type matches that of the set. This is not the case with the other individual. Since the type  $\perp$  is a subtype of every type in the type hierarchy, nothing commits us to believe that  $h$  is a member of the set. More formally,  $h: \text{president} \notin ||\text{president}||$ , for  $h: \text{president}$  is not equal to  $h: \perp$  by the definition of  $\equiv$ . Also, a negative answer is obtained for the question, *Is Hilary Clinton a popular president?* Because  $h: \text{president} \notin ||\text{popular}|| \cap ||\text{president}||$ . But affirmative answers to the questions *Is Hilary Clinton a potential president?* or *Is Hilary Clinton a potential official?* are always obtained.

Thus, by assigning the type  $\perp$  to the adjective *potential* we were able to represent the non-committal membership notion formally. An ad hoc approach would require a non-compositional, non-practical hand-coding of phrases such as *presumed dead*, *alleged thief*, and so on.

## 7 Comparison with Other Approaches

We will compare our approach to CN analysis of modification with that of Montague’s and Davidson’s approaches. In Montague semantics, modifiers are syntactically analyzed as functions from predicates to predicates, and semantically as functions from predicate intensions to predicate extensions. This approach, in line with Montague’s doctrine of generalizing to the hardest case, provides a uniform treatment of modification. However, this generalization, as [14] notes, “blurs” the fact that when an extensional modifier is applied to a predicate the compound is interpreted as the conjunction of the two predicates<sup>8</sup>, as in (a). For work regarding CNs within Montague’s framework, see [22] and [6].

In the Davidson approach, as in the work of [15], on the other hand, the constituents of a CN are analyzed as conjunctions of a number of predicates, whose domains range over the set of individuals. Such an analysis is simple and straightforward with regard to extensional modifiers such as *red* and event-indicating, non-extensional modifiers such as *former*. Thus, the expressions in (a) and (c), from [15], are, respectively, represented as in (b) and (d):

- a) That is a red rose                      b)  $\exists x \text{ red}(x) \wedge \text{rose}(x)$   
 c) Jerry is a former president      d)  $\exists e[ \text{presidency}(e) \wedge \text{Theme}(\text{jerry}, e) \wedge \text{former}(e) ]$

As these examples show, predicates are of first-order nature, which is an advantage over Montague’s, which invokes, as seen by some people, ontologically dubious notions such as possible worlds. However, things get complicated for the Davidsonian

<sup>8</sup> This conjunctive reading can be accounted for with additional semantic rule to the grammar.

analysis when non-committal modifiers are involved. It is difficult to see how a uniform approach can be obtained, where the syntactic and semantic functions stay in close correspondence as compositionality demands and as demonstrated in Montague's. Thus, uniformity is seriously compromised by simplicity.

Our approach, on the other hand, avoids the difficulty encountered by the two approaches. We obtained uniformity not by generalizing to the hardest case, as Montague did, but to the simplest case, conjunction or intersection. This, of course, has been achieved at the expense of adding types to both sets and their elements. Thus, in our approach, as in Davidson, (a) is interpreted as conjunction of the modifier *red*, given the type  $\top$ , and the noun *rose*. Also, as demonstrated in (6.5), we were able to represent non-subjective modifiers; something that Davidson's approach has difficulty accounting for. This is, of course, in addition to our approach's capability of accommodating bracketed expressions and its machine-implementability. The latter is the case since our predicates, as in Davidson's approach, are first-order with some extra overhead computation to work out types.

## 8 Conclusion

We have presented a unified semantic approach to complex nominals. We started by providing an analysis of the ontological nature of the constituents of complex nominals, the adjectives and common nouns. We argued that typed sets are capable of capturing the semantic aspects of extensional and intensional terms, using first-order predication. We argued that typed sets can, without resorting to notions such as possible worlds, show that co-extensive terms have different meanings. Then, we presented the rules and axioms, as well as typing rules, for handling complex expressions. We demonstrated the new approach by applying it to several examples. In particular, we showed how typed set could capture the semantic values of otherwise hard to represent terms such as *presumed*. We then compared our approach to the way CNs are analyzed within two well studied semantic frameworks, Montague and Davidson. We argued that our approach is as general as Montague and is as first-order-predication-oriented as Davidson. We argued that this feature is of significant importance for machine-implementability.

## References

1. Abdullah, N., and Frost, R.: Adjectives: a Uniform Semantic Approach, the 18th Canadian Conference on Artificial Intelligence, Eds. Kégl, B., Lapalme, G.: LNAI 3501, Springer-Verlag (2005): 330-341
2. Alshawi, H.: Memory and Context for Language Interpretation. Studies in Natural Language Processing. Cambridge University Press, Cambridge, England. (1987)
3. Bergsten, N.: A Study on Compound Substantives in English. Almquist, 1991.
4. Costello, F. J.: Investigating creative language: People's choice of words in the production of novel noun-noun compounds. In Proceedings of the 24th Annual Conference of the Cognitive Science Society, 2002.
5. Downing, P: On the Creation and Use of English Compound Nouns. Language 53. 810-842, (1977)

6. Dowty, D: Word Meaning in Montague Grammar. D. Reidel Publishing Co., Do, (1979)
7. Fan, J., Arker, K. and Porter, B. W.: The knowledge required to interpret noun compounds. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, (2003)
8. Fodor, J. & Lepore, E.: The pet fish and the red herring: why concepts aren't prototypes, *Cognition* 58, pp. 243–276, (1996)
9. Guarino, N., Carrara, M. and Giaretta, P.: An Ontology of Meta-Level Categories. In: *Knowledge Representation: (1994)* 270-280
10. Hirst, G.: Semantic interpretation and ambiguity. In *Artificial Intelligence (An International Journal)* Vol 34 PP 131-177, (1988)
11. Hobbs, J, Stickel, M.: Douglas Appelt, and Paul Martin. Interpretation as abduction. *Journal of Artificial Intelligence*, 63(1-2):69-142. (1993)
12. Jespersen, O.: A modern English grammar on historical principles. In: G. Alien, Unwin Ltd. (Eds.), London, pp. 1909–1949. (1954)
13. Kamp, H., and Partee, P.: Prototype theory and compositionality. *Cognition* 57, 129-191. (1995)
14. Lappin, S.: An Introduction to Formal Semantics. In M. Aronoff and J. Rees-Miller, editors, *The Handbook of Linguistics*, pages 369-393, Blackwell, Oxford. (2000)
15. Larson, R.K.: The grammar of intensionality. In: Preyer, G. (ed.): *On Logical Form*, Oxford University Press. (2002)
16. Lees, R.: Problems in the Grammatical Analysis of English Nominal Compounds. In Bierwisch and Heidolph (eds.) *Progress in Linguistics*, The Hague. (1970)
17. Levi, Judith N.: *The Syntax and Semantics of Complex Nominals*. Academic Press. (1978)
18. Lynott, D., and Keane, M.: The comprehension of novel noun-noun compounds: The influence of out-of-context interpretations on in-context understanding. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society*. (2002)
19. Marchand, H.: *The Categories and Types of Present Day English Word Formation*. C.H Becksche, Munich. (1969)
20. Partee, B. H.: Privative adjectives: subsective plus coercion. In Zimmermann, T.E. (ed.): *Studies in Presupposition* . (2001)
21. Raskin, V., and Nirenburg, S.: Lexical Semantics of Adjectives: A Microtheory of Adjectival Meaning, in: *MCCS-95-288*, Las cruces, N.M.: New Mexico State Univ.(1995)
22. Siegel, M.: Capturing the adjective. PhD dissertation, University of Massachusetts. (1976)

# A Hybrid Approach to Improving Automatic Speech Recognition Via NLP

Kimberly Voll

School of Computing Science  
Simon Fraser University  
8888 University Drive  
Burnaby, BC V5A1S6  
kvoll@cs.sfu.ca

**Abstract.** In many domains, automated speech recognition (ASR) demands highly robust and accurate recognition software. Unfortunately, in such domains, even a 99% accurate recognizer is inadequate, and other methods for increasing the reliability and performance of ASR must be considered. As a possible solution to this problem, post-speech-recognition error detection can assist in proofreading more efficiently. To this end, we have developed a multi-heuristic algorithm using natural language processing to detect recognition errors. As a proof of concept, we have applied this algorithm to the radiology domain. The results are encouraging, showing a 22% increase in the recall performance, and a 6% increase in the precision performance, over the best individual technique.

## 1 Introduction

The increasing accuracy of automated speech recognition (ASR) technology, as well as the many benefits, have prompted the introduction of ASR systems in a wide variety of venues. For example, in medicine ASR can offer improved patient care and resource management in the form of reduced report turnaround times, reduced staffing needs, and the efficient completion and distribution of reports [1,2]. In some domains, however, the accuracy remains too low, making ASR a poor choice. This becomes apparent if we consider that even a supposedly 99%-accurate speech recogniser still averages one error out of every hundred words<sup>1</sup>, with no guarantees as to the seriousness of such errors<sup>2</sup>.

While directly increasing ASR accuracy beyond 99% is not likely in the near future, we can improve matters through post-ASR error detection. By applying natural language processing techniques, in the form of heuristics, we can create a system that quickly identifies ASR errors (and ultimately corrects them), streamlining the process of proofreading and restoring much of the efficiency of ASR.

Current research in post-ASR error detection has been applied mostly to conversational systems. Statistical methods such as word co-occurrences [3,4,5]

---

<sup>1</sup> When considering the standard word-error rate (WER).

<sup>2</sup> The average accuracy in a domain such as radiology is typically between 80-85%.

are popular since ASR errors “are found to occur in regular patterns rather than at random” [6]. Unfortunately, the variety of potential recognition errors is inadequately captured by statistical methods alone. Hybrid methods, however, attempt to complement the weaknesses of an individual approach with the strengths of others. Thus, we have developed a hybrid method for post-recognition detection that uses both statistical and non-statistical techniques. By employing both techniques in the form of multiple heuristics, it is possible to overcome many of the inherent limitations of single techniques, while detecting errors of any type. As a proof of concept, we implement the hybrid, post-ASR detection method in the domain of radiology reporting.

### 1.1 On the Nature of Recognition Errors

As outlined in Kukich [7], there are five *levels of text-based errors*, namely lexical/structural<sup>3</sup>, syntactic, semantic, discourse, and pragmatic.

In addition, there are six *recognition error types* that can cause errors at some of these levels:<sup>4</sup>

**Stop Word Errors.** Any error involving a stop word (i.e. words with low semantic load, such as prepositions or determiners). In general, stop words can result in errors at the syntactic or semantic level.

**Merge Errors.** Two or more words are erroneously recognised as a single word [8]. E.g. “wreck a nice” → “recognise”.

**Split Errors.** A single word erroneously recognised as two or more words [8]. E.g. “recognise” → “wreck a nice”.

**Substitution Errors.** The replacement of one word by another [9].

**Insertion Errors.** The insertion of a word that is not part of the original utterance [9].

**Deletion Errors.** A word in the original utterance that does not appear in the final ASR output.

Deletion errors are difficult to detect as words typically leave little record of their absence. Similarly, the detection of stop word errors is also difficult due to their prevalence in the language and the small semantic role they play. Consequently, many error detection systems focus on the remaining four error types. By developing a hybrid method, however, we have been able to integrate weaker methods—that detect deletions but not much else—with stronger methods for detecting other error types, thus creating a wide-coverage error-detection system for assigning post-recognition confidence scores to a text.

<sup>3</sup> Depending on the domain, the misrecognition of specially formatted lexical items may arise. This is frequently seen in the interpretation of radiology reports where complex lexical items such as “L4/5” are misinterpreted as “L for/five”. To represent such instances, we have introduced the error level, “structural”, that sits parallel to the lexical level.

<sup>4</sup> Note that it is impossible for the recogniser to introduce errors at the discourse or pragmatic level (except those which may follow as side effects of other errors) since no recogniser-based processing occurs at these levels. Furthermore, since all recognised words are produced from a pre-defined lexicon, lexical errors are not possible.

## 2 Error Detection in Restricted Domains

When working in a limited domain, such as radiology, one can identify features specific to radiology reports. For example, by examining the context of the various sections of a standard radiology report, certain features emerge that represent the expected features of words occurring in a particular section. Thus, a “Procedures” section will contain those concepts relating to radiological procedures with a higher probability than those relating to other areas. Similarly, if a report is discussing an examination of the knee, concepts relating to the other parts of the body will have a lower probability. When these and other heuristics are combined together it is possible to generate a characterization of the “correctness” of a particular word or phrase in the report.

## 3 A Hybrid Approach to Error Detection

By observing that the individual heuristics each detect a limited subset of the error types listed above, when combining them together it is possible to create a hybrid error-detection algorithm that covers all error types. Our intent with the hybrid method is to complement the weaknesses of one approach with the strengths of another.

**Objective.** To develop a mapping from the individual words of a dictation to a confidence score or error tag set.

To achieve this mapping, the desired system must have some means for assigning confidence scores and ultimately identifying recognition errors within in a text. This requires the identification of features whose values can be combined to form a single confidence ranking for a word or phrase. In this way it is possible to define different error-detection algorithms that may rely on different sets of features.

The ultimate goal of an error-detection system is a mapping that, when applied to a text such as a radiology report, will output a list of errors detected. This list can be expressed superficially as a tag indicating a word is “correct” or “incorrect”, where “incorrect” means that a word can be described according to one of the error types outlined in Section 1.1. Thus, all words are mapped to the error tag set,  $\{correct, incorrect\}$  (irrespective of their error type).

In a hybrid method, however, this mapping relies on the interaction of the various error-detection heuristics. There are two possibilities for arriving at the word-level tag map. In the *direct method*, the indication of an erroneous word by at least one heuristic is sufficient to trigger an “incorrect” tag on that word. Thus individual confidence scores are evaluated on the basis of a threshold and the combined result of each error-detection heuristic is a binary tag (as above). In the *indirect method*, the output from each heuristic is taken as input to a meta-level heuristic such that each word is provided with a confidence score based upon the weighted aggregation of any scores from the individual heuristics. As an initial proof of concept, we employ the direct method in our calculations.

**Observation 1.** No single error-detection technique is sufficient to detect all potential errors in a dictation.

The goal in any post-recognition error-detection system is 100% coverage of all error types and 100% accuracy in identifying errors. While the various statistical and non-statistical methods of error-detection are each sensitive to a particular subset of error types, none provide complete coverage, nor have any implementations achieved 100% or near-100% accuracy [3,10,6,5,8]. In some cases, such as the use of stop lists<sup>5</sup> in statistical techniques, complete coverage is impossible.

**Observation 2.** By combining those methods of error detection that are complementary in their coverage of error types, it is possible to achieve greater sensitivity to errors within radiology reports.

Although individual error-detection techniques may be insufficient, if their coverage of error types is shown to be complementary, then the combination of multiple heuristics will result in a higher coverage of error types. In addition, overlapping areas of coverage will increase the reliability of the final confidence score or error mapping.

**Conclusion.** A hybrid approach is the best choice for post-recognition error-detection.

## 4 The Error Detection System

To demonstrate the efficacy of the proposed hybrid, post-ASR methodology, we present a proof of concept in the radiology domain with three component heuristics: two statistical heuristics using word-occurrence probabilities, namely co-occurrence relations and pointwise mutual information (PMI), and one non-statistical heuristic, a constraint-handling-rules parser.

### 4.1 Materials

**Corpora.** The Canada Diagnostic Centre (CDC) in Vancouver, BC, has provided over 2700 corrected and de-identified radiology reports (using the Dragon NaturallySpeaking speech-recognition system, version 7.3) as well as an additional corpus of 30 raw, uncorrected radiology reports paired with their corrected versions.

Out of these test reports, there is an average of 11.9 errors per report, with an average report length of 80.8 words. This represents an average word-error rate (WER) of 15%.

### 4.2 Methods

To find the actual errors in our test reports, we align the corrected and uncorrected reports, determine any differences and tag them as errors. We then

---

<sup>5</sup> A list of words excluded from an analysis.



compare these to the flagged errors from our program output to obtain our results: a match is considered a correct detection, or *true positive*; a flagged error that does not correspond to an actual error is considered a *false positive*; an error not flagged is considered a *false negative*.

In calculating all results, *Recall* is a measure of the number of errors correctly detected over the total number of errors actually present; *Precision* is a measure of the number of errors correctly detected over the total number detected.

As a final note, all tools were designed and run on a Mac G4, 1.5 GHz, OS X 10.3.9.

## The Statistical Heuristics

**Observation 4.** The probability of a misrecognised term is lower than the probability of a correctly recognised term.

We used two probabilistic heuristics: co-occurrence relations and PMI. Underlying both is the notion that in identifying patterns common to error-free reports, we can automatically detect inaccuracies within novel reports. By choosing two statistical algorithms, the results can be combined to smooth out any anomalies within the calculations themselves to produce more reliable results, as well as to provide a comparative evaluation of the two techniques.

As part of the setup for both co-occurrence analysis and PMI, the co-occurrence statistics of varying window sizes have been compiled for the 2700, anonymised MRI reports.<sup>6</sup>

The first heuristic is based on the work in Voll [11,12] in which co-occurrence relations [4,13,10] were found to have a high recall in detecting errors in radiology reports. Given a sufficiently representative training corpus, we can associate words with particular contexts based on that corpus. We can then apply these word-context statistics to determine the probability of a word occurring in a given context in a report. This probability represents a measure of the confidence of that word; if it falls below a certain threshold the word will be flagged as a possible error.

For each uncorrected report we determine the context of each word, calculate the co-occurrences and apply the appropriate collection of co-occurrence statistics from the training data.

Given a word,  $w$ , occurring in a document,  $d$ , a context window,  $C(w, d, n)$ , is defined as the  $n$  words occurring to either side of  $w$  in  $d$ .<sup>7</sup>

<sup>6</sup> It is important to note that in co-occurrence analysis, stop words are usually omitted since their overabundance in a text can affect the resulting probabilities disproportionately. This immediately limits the maximum error detection.

<sup>7</sup> For example, consider the incorrect sentence fragment in Sentence 1:

...possible spondylolysis eye laterally of L5... (Sentence 1)

We can generate the following co-occurrences for the target word, “eye”, with a context window of 2 (up to 2 words to either side of “eye”, ignoring stop words): {eye possible, eye spondylolysis, eye laterally, eye L5}.

Using Bayes' Theorem (Equation 1), we then combine the probability of each word that occurs within the context window of the target word, and the probability of the target word itself.

$$P(t|C) = \frac{P(t) * P(C|t)}{P(C)} \quad (1)$$

In general, we use the following calculation for each word (where *win* represents the size of the context window being used):

$$\frac{P(w_i|w_{i-win}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+win})}{P(w_i)P(w_{i-win}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+win})} \quad (2)$$

While Bayes' Theorem is a straightforward approach to determining the probability of a word given its context, other methods of aggregation could be easily applied here.

The second heuristic is based on the work of Inkpen and Désilets who suggest similar results using PMI. They also discuss other techniques previously employed, but conclude that PMI performs the best, in part because of the potential to scale up well to larger databases (which is ultimately desired for better characterization of radiology reports) [14,15]. Like the co-occurrence method above, given a sufficiently representative training corpus, it is possible to derive word probabilities based on the probability of occurrence within that corpus. Similarly, the probability of a word co-occurring with another word within a particular context window can be determined by the frequency of such a co-occurrence within the training corpus. The probability of two words occurring independently versus the rate at which they occur together, provides a measure of independence that can be used as a measure of the likelihood of a word occurring in a given context in a report. If that probability falls below a certain threshold the word will be flagged as a possible error.

As described in Inkpen and Désilets [14], a semantic similarity score between two words,  $w_1$  and  $w_2$  is based on the shared information load of both words. Equation 3 shows the calculation of PMI for two words. Here  $C(w_1, w_2)$ ,  $C(w_1)$  and  $C(w_2)$  represent the frequency of occurrence (in the training corpus) while  $n$  is the total number of words in the corpus [14]. Thus the PMI semantic similarity measure is a reflection of probability of two words occurring together, where "together" is limited by the defined context-window size, and the individual probability of each word occurring in the training corpus [14].

$$PMI(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1) \cdot P(w_2)} = \log \frac{C(w_1, w_2) \cdot n}{C(w_1) \cdot C(w_2)} \quad (3)$$

The basic PMI calculation in Equation 3 is applied to two individual words. In the case of a document, however, the desired outcome is the semantic similarity of an individual word to the context in which it occurs.

### Results

In both heuristics, to find the actual errors in our test reports, we align the corrected and uncorrected reports, determine any differences, and tag them as errors. We then compare these to the flagged errors from our program output to obtain our results, where “true positive”, “false positive” and “false negative” are defined as in Section 5.2.3.

The system is able to identify error candidates in under a minute in all cases, underscoring its viability for real-time use<sup>8</sup>.

The results obtained are shown in Table 1 while the results for the PMI calculation on our test corpus are shown in Table 2. *Corpus Size:Training* is the the number of reports in the training set, while *Corpus Size:Test* is the number of test cases on which the system was run. In the Sensitivity, “Salient Words Only” means errors involving words with little or no semantic load (such as prepositions) were not considered.

**Table 1.** Co-occurrence analysis on speech-recognised reports with windowsize=1, threshold=0

Sensitivity	Accuracy			Corpus Size	
	Recall	Precision	f-measure	Training	Test
All Errors	52%	36%	42%	2751	30
Salient Words Only	86%	29%	44%	2751	30

**Table 2.** PMI analysis on speech-recognised reports with windowsize=10, threshold=100

Sensitivity	Accuracy			Corpus Size	
	Recall	Precision	f-measure	Training	Test
All Errors	35%	34%	34%	2751	30
Salient Words Only	73%	35%	47%	2751	30

*Discussion.* The recall reflects a high sensitivity to errors and a low rate of false negatives when only salient words are considered. This is especially important in medicine as errors missed could have serious ramifications. In contrast, the precision is low, indicating a high rate of false positives. Although still important overall, false positives are nonetheless identifiable by the radiologist and do not affect report quality. In most cases these false positives are generated by word-context pairs that were not previously encountered in the training data. Thus we have  $P(C|T) = 0$ , which results in  $P(T|C) = 0$  by Equation 1. By increasing the number of reports in the training corpus, however, we can ensure a greater

<sup>8</sup> There is a one-time overhead cost associated with generating the co-occurrence statistics for the training sets. Once generated, however, the database is simply stored and referenced. Re-generation would only occur if new training data were added.

coverage of the terms that typically occur in a radiology report. This will cause the rate of false positives to drop and improve the precision. Although the ideal training corpus would contain every possible context of every possible word in a radiology report, radiology nonetheless does not exhibit a wide variation within reports. A fairly accurate depiction of the possible patterns within a report is feasible with a large enough training set. Interestingly, though, some false positives may be advantageous, indicating rare occurrences that merit closer inspection by the radiologist to ensure there are no mistakes.

The rate of error detection, or filtering, is affected by the threshold value,  $K$ . Higher values of  $K$ , mean less filtering and a higher WER, while lower values of  $K$ , mean greater filtering and a lower WER. In this way it is possible to increase the recall level to near 100%, however, there is a corresponding loss of precision. Nonetheless, this does allow for some flexibility in balancing between the recall and precision measurements<sup>9</sup>.

Note that the PMI results shown here do not reflect the same degree of success that was seen in Inkpen and Désilets [14]. This is likely a reflection of the significantly smaller training set used. If a word is not found in the training data, then its probability and the probability of it occurring in any co-occurrence tuples will be zero, resulting in an incalculable PMI value. By default, the system sets these values to zero, indicating no semantic similarity.

## Syntactic Analysis

**Observation 5.** The points of failure in a syntactic parse can be used to identify likely error candidates.

Although statistical methods have dominated error detection in ASR, their use of stop lists and surface-level analysis prevents such systems from achieving 100% accuracy. To fill this gap, non-statistical methods can offer a more in-depth analysis of the features within a text.

Syntactic recognition-errors include words or phrases that are out-of-place with respect to their syntactic placement. In a misrecognised text, for instance, a verb may occur in the text where the syntactic analysis would predict a noun. It is possible to identify these syntax errors and apply a weight to determine a confidence score for words within the text. Thus, as a component of the hybrid approach to error detection, a syntactic parser can be used to identify syntactic errors, including those which involve stop words and deletions.

With this in mind, a parser was developed to analyse radiology reports. In the interest of rapid prototyping sufficient for proof of concept, the parser was built upon a constraint handling rules grammar, or CHRg [16] and inspired by Property Grammars [17]. Unique to property grammars, the properties defining the allowable constructs within the grammar can be tagged as “relaxable” [18].

---

<sup>9</sup> The decision of the threshold value was one of trial and error. In the end, keeping the threshold at zero (i.e. no effect on the amount of error filtering) gave us the best results in light of the already low precision scores. If a larger training corpus improves the precision score, a more appropriate threshold could be chosen.

While needing to relax a property is likely to indicate an error (i.e. an incorrect term or an incomplete phrase), the parse is able to continue and information regarding the nature of the error is collected (i.e. those properties that were not met). The result is a robust parser that can detect and locate errors within the text. What’s more, by characterizing the grammar as a series of properties, the properties constraining the language within radiology reports are easily captured.

During the parse, a series of property checks are performed determine the phrases and the type of each phrase. Each phrase type has its own rule set defining its specific properties. If those properties are met (or labeled “relaxable”) then a constraint representing a constituent of that phrase type is added to the constraint store. Otherwise the phrase is labeled and added to the constraint store as “unknown”.

The grammar is comprised of constraint handling rules describing the combination of constituents when certain constraints are observed in the constraint store. After each change in the constraint store, these constraint handling rules are consulted and whenever possible constraints are combined to form new constraints which are then used to update the constraint store. In this way, the parse is completed conjoining sub-phrases wherever permitted by the constraint definitions. When no further changes are possible, the system is “settled” and the current contents of the constraint store are output. During the parse, the system maintains a list of all “unknown” constituents. These are output separately at the end of the parse.

The interpretation of the results for error detection is currently performed manually. Errors can take three forms given the parser output: phrases tagged as “unknown”, unsatisfied property lists, and incomplete parse segments. “Unknown” tags represent words or phrases that could not be assigned a phrase type by the parser (or were not recognized by the pre-processor).

### Results

**Table 3.** CHR parser results on all error types

Report Type	Accuracy			Corpus Size
	Recall	Precision	f-measure	Test
All Errors	29%	34%	32%	30
Syntactic Errors Only	71%	17%	27%	30

The drop in precision in the second set is because the total number of errors returned by the parser is the same, regardless of error type being considered. In Table 3 this caused a drop in the precision since a smaller number of errors were considered “correct”, that is actual syntactic errors. In addition, while the parser takes longer than the statistical techniques (up to three minutes in some cases), there is no overhead cost associated with generating the co-occurrence statistics. Also, in all cases the slow run time was attributable to the preliminary nature of the parser and will improve with future iterations.

*Discussion.* Since a syntactic parser is capable only of finding errors relating to the syntax of the input text, it naturally performs poorly when measured against all errors within a report. However, when limited to only the syntactic errors, we see that the system performs well above chance, motivating continued research.

5 A Hybrid Approach

As a proof of concept of the proposed hybrid error-detection method, the above heuristics have been applied in combination to the test corpus. The results in Table 4 do show that post-recognition error-detection can improve speech recognition output. Furthermore, the combined application of these heuristics shows a 22% increase over the best single heuristic technique, favouring the hybrid method over previous, independent applications of error-detection methods in ASR when applied to radiology reports. In addition, since these reports were part of an ongoing collection by the CDC, they were produced only when time was available, and include all scan types, unlike the training data, which is limited to MRI. Thus, the results shown here show extensibility beyond the training data to broader divisions of reports.

An important aspect of this analysis is the omission of stop words, or low-information-bearing words. These words are ignored because it is often observed that a mis-recognised stop word rarely entails a shift in the intended semantics. Exceptions exist, however, such as a substitution of “and” for “at the”, that may have more serious consequences in medicine, and may prove difficult for human editors to detect. As a result, a more detailed analysis of stop words is currently undergoing.

**Table 4.** Results from the individual algorithms on all errors (and all words) and the combined results on all errors

Report Type	Accuracy			Corpus Size	
	Recall	Precision	f-measure	Training	Test
Best Co-Occur	52%	36%	42%	2751	30
Best PMI	35%	34%	34%	2751	30
Parser	29%	34%	32%	2751	30
Combined	74%	42%	54%	2751	30

6 Future Work

This research is ongoing, including experiments with innovative, non-stochastic methods that rely on syntactic as well as semantic analysis of the text itself, such as conceptual similarity [19,20]. Such techniques will make it possible to evolve beyond just detection to the much harder problem of automated correction. This can include semi-automated correction, whereby the radiologist is presented with intelligent suggestions for correction of recognition errors.

Since the error coverage differs with each error-detection method, not all words may have numerical scores assigned from all algorithms. If the output from each algorithm is a measure of confidence in the recognizer output, then the combined result of applying all heuristics via the hybrid algorithm to the text will result in a complex confidence score for each word. These results can be combined beyond the naïve *direct method*, with the choice of meta-level heuristic affecting the final confidence rankings and thus the overall performance of the system.

Finally, this technique could easily be extended to other areas of medicine that share the same properties of restricted vocabulary seen in radiology, provided an adequate training corpus is available.

## 7 Conclusion

Despite the trend toward automated speech recognition, in some domains the accuracy of ASR remains a limiting factor. By incorporating a hybrid approach to post-recognition error detection using NLP, however, these errors can be efficiently detected, restoring the benefits of ASR in many cases.

## Acknowledgements

We wish to thank the Canada Diagnostic Centre for their continued support of this research endeavour, as well as Simon Fraser University and the NSERC Discovery Grant Program.

## References

1. Horii, S., Redfern, R., Kundel, H., Nodine, C.: PACS technologies and reliability: Are we making things better or worse? In: Proceedings of SPIE. Volume 4685. (2002) 16–24
2. Mehta, A., Dreyer, K., Schweitzer, A., Couris, J., Rosenthal, D.: Voice recognition – an emerging necessity within radiology: Experiences of the massachusetts general hospital. *Journal of Digital Imaging* **11**(4) (1998) 20–23
3. Jeong, M., Kim, B., Lee, G.: Using higher-level linguistic knowledge for speech recognition error correction in a spoken Q/A dialog. In: Proceedings of the HLT-NAACL special workshop on Higher-Level Linguistic Information for Speech Processing. (2004) 48–55
4. Jurafsky, D., Martin, J.: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall Inc (2000)
5. Allen, J.F., Miller, B.W., Ringger, E.K., Sikorski, T.: A robust system for natural spoken dialogue. In: Proceedings of the 34th Annual Meeting of the ACL. (1996) 62–70
6. Kaki, S., Sumita, E., Iida, H.: A method for correcting errors in speech recognition using the statistical features of character co-occurrence. In: ACL-COLING. (1998) 653–657

7. Kukich, K.: Techniques for automatically correcting words in text. *ACM Computing Surveys* **24**(4) (1992) 377–439
8. Ringger, E.K., Allen, J.F.: A fertility model for post correction of continuous speech recognition. In: *ICSLP96*. (1996) 897–900
9. Gillick, L., Ito, Y., Young, J.: A probabilistic approach to confidence measure estimation and evaluation. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing*. (1997) 879–882
10. Sarma, A., Palmer, D.: Context-based speech recognition error detection and correction. In: *Proceedings of the HLT-NAACL 2004*. (2004) 85–88
11. Voll, K., Atkins, S., Forster, B.: Improving the utility of speech recognition through error detection. In: *SCAR Annual Meeting*. (2006) in press.
12. Voll, K.: A Methodology of Error Detection: Improving Speech Recognition in Radiology. PhD thesis, Simon Fraser University, School of Computing Science, 8888 University Drive, Burnaby, BC, Canada (2006)
13. Manning, C.D., Schütze, H.: *Foundations of statistical natural language processing*. MIT Press (2002)
14. Inkpen, D., Désilets, A.: Semantic similarity for detecting recognition errors in automatic speech transcripts. In: *Proceedings of EMNLP, Vancouver, British Columbia, Canada, Association for Computational Linguistics* (2005) 49–56
15. Turney, P.D.: Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In: *Proceedings of the Twelfth European Conference on Machine Learning, Freiburg, Germany* (2001) 491–502
16. Christiansen, H.: CHR grammars. *Theory and Practice of Logic Programming* **5**(4) (2005) 467–501
17. Blache, P.: Property grammars: A fully constraint-based theory. In Christiansen, H., Skadhauge, P.R., Villadsen, J., eds.: *Constraint Solving and Language Processing*. Volume 3438 of *Lecture Notes in Artificial Intelligence*. Springer (2005)
18. Dahl, V., Voll, K.: Concept formation rules: An executable cognitive model of knowledge construction. In: *Proceedings of the First International Workshop on Natural Language Understanding and Cognitive Sciences, Porto, Portugal* (2004) 28–36
19. Caviedes, J.E., Cimino, J.J.: Towards the development of a conceptual distance metric for the UMLS. *Journal of Biomedical Informatics* **37** (2004) 77–85
20. Shiffman, S., Detmer, W.M.S., Lane, C.D., Fagan, L.M.: A continuous-speech interface to a decision support system: I. Techniques to accommodate misrecognized input. *AMIA* **2** (1995) 36–45



# Planning in Multiagent Expedition with Collaborative Design Networks

Y. Xiang and F. Hanshar

University of Guelph, Canada

**Abstract.** DEC-POMDPs provide formal models of many cooperative multiagent problems, but their complexity is NEXP-complete in general. We investigate a sub-class of DEC-POMDPs termed *multiagent expedition*. A typical instance consists of an area populated by mobile agents. Agents have no prior knowledge of the area, have limited sensing and communication, and effects of their actions are uncertain. Success relies on planing actions that result in high accumulated rewards. We solve an instance of multiagent expedition based on collaborative design network, a decision theoretic multiagent graphical model. We present a number of techniques employed in knowledge representation and demonstrate the superior performance of our system in comparison to greedy agents experimentally.

## 1 Introduction

Decentralized partially observable Markov decision processes (DEC-POMDPs) (e.g., [1]) extend POMDPs to multiagent systems. DEC-POMDPs provide formal models of many cooperative multiagent problems. However, in general, their complexity is nondeterministic exponential time complete (NEXP-complete) [2].

We consider a sub-class of DEC-POMDPs which we term as *multiagent expedition*. A typical instance consists of a large area populated by objects as well as mobile agents. Activities of agents include moving around the area, avoiding dangerous objects, locating objects of interests, and manipulating objects in various ways depending on the nature of application. The effect of an action is generally uncertain. Agents have no prior knowledge on the area. That is, they do not know, *a priori*, where dangerous or interesting objects are located. Instead, they try to identify nearby objects based on limited sensing of the local environment. Successful manipulation of an interesting object sometimes requires proper actions of a single agent and sometimes requires cooperation of multiple agents through limited communication. The success of an agent team depends on the number of objects successfully manipulated as well as the quality of each manipulation. Practical examples of multiagent expedition include undersea adventure, planet expedition, disaster rescue, anti-air defense, etc. In this paper, we specify precisely the instance of multiagent expedition used in this investigation.

Our knowledge representation is based on collaborative design networks (CDNs) originally proposed [10,11] as a decision-theoretic framework for multiagent, optimal industrial design. The expressive power of the framework, however,

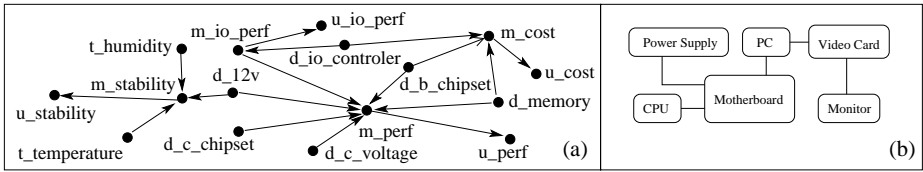
goes beyond design. As long as domain dependencies can be encoded into sparse and distributed graphical structures, the framework supports autonomous, optimal, and efficient multiagent decision making. To further explore its generality, this work investigates the application of CDNs to multiagent expedition.

Section 2 introduces background on CDNs. The instance of multiagent expedition that we investigate is specified in Section 3. How we approach this generally intractable problem computationally is detailed in Section 4 and our experimental results are reported in Section 5. Additional related work is discussed in Section 6 after presenting ours, to facilitate comparison.

## 2 Background on CDNs

We briefly review background on CDNs, whose full technical details can be found in [10,11]. CDNs are motivated by collaborative industrial design in supply chains. An agent responsible for the design of a component encodes its design knowledge and preference into a *design network* (DN)  $S = (V, G, P)$ . The *domain* is a set of discrete variables  $V = D \cup T \cup M \cup U$ , where  $D, T, M, U$  are disjoint.  $D$  is a set of *design parameters*.  $T$  is a set of *environmental factors* (working conditions) of the product under design.  $M$  is a set of objective *performance measures* and  $U$  is a set of subjective *utility functions* of the agent's principal.

The dependence *structure*  $G = (V, E)$  is a directed acyclic graph (DAG) whose nodes are mapped to elements of  $V$  and whose set  $E$  of arcs is from the following legal types: Arc  $(d, d')$  ( $d, d' \in D$ ) signifies a design constraint. Arc  $(d, m)$  ( $m \in M$ ) represents dependency of performance on design. Arc  $(t, t')$  ( $t, t' \in T$ ) represents dependency between environmental factors. Arc  $(t, m)$  signifies dependency of performance on environment. Arc  $(m, m')$  defines a composite performance measure. Arc  $(m, u)$  ( $u \in U$ ) signifies dependency of utility on performance. Fig. 1 (a) shows a trivial DN.



**Fig. 1.** (a) A trivial DN for PC motherboard. First letter of a node label indicates its type. (b) Hypertree of a simple CDN.

$P$  is a set of potentials, one for each node  $x$ , and each is formulated as a probability distribution  $P(x|\pi(x))$ , where  $\pi(x)$  is the parent nodes of  $x$ , but its semantics depends on  $x$ .  $P(d|\pi(d))$  encodes a design constraint.  $P(t|\pi(t))$  and  $P(m|\pi(m))$  are typical probability distributions. Each utility variable has the domain  $\{y, n\}$ .  $P(u = y|\pi(u))$  is a utility function  $u(\pi(u))$  whose values range in  $[0, 1]$ .  $P(u = n|\pi(u))$  is assigned  $1 - P(u = y|\pi(u))$ . Each node  $u$  is assigned a weight  $k \in [0, 1]$  such that weights of all utility nodes sum to one.

With  $P$  thus defined,  $\prod_{x \in V \setminus U} P(x|\pi(x))$  is a joint probability distribution over  $DUTUM$ . With the assumption of additive independence among utility variables, the expected utility of a design  $\mathbf{d}$  is  $EU(\mathbf{d}) = \sum_i k_i (\sum_{\mathbf{m}} u_i(\mathbf{m}) P(\mathbf{m}|\mathbf{d}))$ , where  $\mathbf{d}$  (bold) is a configuration of  $D$ ,  $i$  indexes utility nodes in  $U$ ,  $\mathbf{m}$  (bold) is a configuration of parents of  $u_i$ , and  $k_i$  is the weight of  $u_i$ .

Each supplier in a supply chain is a designer of the supplied component. Agents, one per supplier, form a collaborative design system. Each agent embodies a design network called a design *subnet* and agents are organized into a *hypertree*: Each hypernode corresponds to an agent and its subnet. Each hyperlink (called *agent interface*) corresponds to design parameters shared by the two subnets (referred to as *public* variables). Hypertree organization specifies to whom an agent can communicate directly. Each subnet is assigned a weight  $w_i$ , representing a compromise of preferences among agents, and  $\sum_i w_i = 1$ . The collection of subnets  $\{S_i = (V_i, G_i, P_i)\}$  forms a CDN. Fig. 1 (b) shows the hypertree of a simple CDN for customized PC design, where the subnet on motherboard is shown in (a).

The product  $\prod_{x \in V \setminus \cup_i U_i} P(x|\pi(x))$  is a joint probability distribution over  $\cup_i (D_i \cup T_i \cup M_i)$ , where  $P(x|\pi(x))$  is associated with node  $x$  in a subnet. The expected utility of a design  $\mathbf{d}$  is  $EU(\mathbf{d}) = \sum_i w_i (\sum_j k_{ij} (\sum_{\mathbf{m}} u_{ij}(\mathbf{m}) P(\mathbf{m}|\mathbf{d})))$ , where  $\mathbf{d}$  is a configuration of  $\cup_i D_i$ ,  $i$  indexes subnets,  $j$  indexes utility nodes  $\{u_{ij}\}$  in  $i$ th subnet,  $\mathbf{m}$  is a configuration of parents of  $u_{ij}$ , and  $k_{ij}$  is the weight associated with  $u_{ij}$ . Through communication along the hypertree, agents can determine the optimal design  $\mathbf{d}^*$  that has the maximum  $EU(\mathbf{d})$  [11].

### 3 The Multiagent Expedition Testbed

The following instance of multiagent expedition is used in our investigation: The area is abstracted as a grid of cells. At any cell, an agent has five possible actions: moving to an adjacent cell along one of four directions (referred to as *north*, *south*, *east*, *west*) or remaining in the current cell (referred to as *halt*). The effect of an action is, however, uncertain. That is, the action *north* may cause the agent to land on each of four unintended cells.

The desirability of an object (located at a cell) is indicated by a numerical *reward*. For simplicity, we abstract *away* the object and associate the reward with the cell. A cell that is neither interesting nor harmful has a reward of a base value. The reward at a harmful cell is lower than the base value. The reward at an interesting cell is higher than the base value and can be further increased through agent cooperation. When a physical object at a given location is to be manipulated (e.g., digging, lifting, pushing, etc.), cooperation is often most effective when a certain number of agents are involved, and the per-agent productivity is reduced when less or more agents are involved. We set the most effective level at 2, although other levels can also be used. For instance, the reward that can be collected by a single agent from a given cell may be 0.3. However, if two agents cooperate and *meet* at the cell, each receives 0.4. If three or more agents meet at the cell, two of them each receives 0.4 and the other

agents receive the base value. This feature promotes effective cooperations and discourages unproductive ones. It is encoded by associating each cell with a reward pair  $(r_1, r_2)$ , where  $r_1$  is the reward collected by a single agent and  $r_2$  is the total reward collected by two cooperating agents. Hence, the above mentioned cell has the reward  $(0.3, 0.8)$ . After a cell has been visited by any agent, its reward is decreased to the base value. As a result of this feature, wandering within a neighborhood will not be productive and agents must move around strategically.

Agents have no prior knowledge about the area on how the rewards are distributed. Instead, at any cell, an agent can perceive the cell's absolute location (e.g., through GPS on Earth or triangulation with two base stations on Mars). It can also perceive the reward distribution in its neighborhood. We set the neighborhood to be the 13 cells shown in Fig. 2 (a), although different settings are also possible. An agent can also perceive the location of another agent if the latter is within a 10 step radius. It can communicate with agents within this radius as well. The objective of agents is to move around the area, cooperate as needed, and maximize the team reward over a finite horizon. They must do so based on local observations and limited interagent communication.

This instance of multiagent expedition is a DEC-POMDP. The state of the environment is described by the location of all agents as well as the distribution of rewards. It is *stochastic* since the effect of actions are uncertain. It is *Markovian* as the new state is conditionally independent of the history given the current state and the joint action of agents. It is *partially observable* because each agent can only perceive its neighborhood, but not the distribution of rewards and agents beyond.

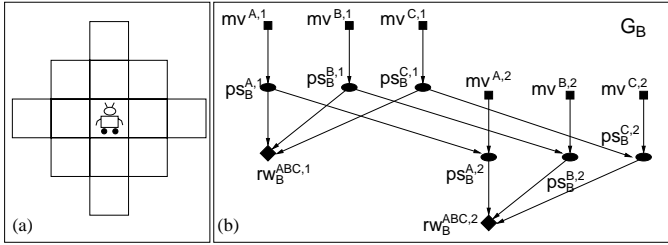
Because an agent can perceive its own location and agents nearby, a significant amount of relevant information in the environmental state is obtained through observation. On the other hand, not all relevant information has been obtained, because knowing the reward distribution beyond the neighborhood will allow the agent to plan better. To capture this difference from the case where agents cannot perceive its own location reliably, we refer to the stochastic process as a decentralized weakly partially observable Markov decision process (DEC-W-POMDP).

## 4 A Decision-Theoretic Graphical Models Approach

Consider the general case of the problem instance with  $n$  agents and horizon  $k$ . Given the current positions of agents, each agent has five possible actions. Hence, there are  $5^n$  joint actions and  $5^{nk}$  joint plans of horizon  $k$ . Since each action has five possible effects, a joint action has  $5^n$  possible effects, a joint plan has  $5^{nk}$  possible effects, and the  $5^{nk}$  joint plans have a total of  $5^{2nk}$  possible effects. Each autonomous agent needs to evaluate these effects, identify the optimal joint plan, and obtain its own optimal action sequence. For six agents and horizon 2, each agent needs to evaluate  $5^{24} \approx 6 \times 10^{16}$  possible effects. To carry out the computation more efficiently, we take the following measures:

**Splitting Agent Team into Groups.** We divide  $n$  agents into smaller groups to allow high inner-group interaction and low inter-group interaction. Grouping has no negative effect on scaling up. It allows group members to stay closely so that they can cooperate effectively, as long as the group size is no smaller than the number of agents to be involved in a most effective cooperation. It allows different groups to stay away from each other, which not only allows the team to explore the area more effectively, but also allows reduction of computation by reducing inter-group interaction. In this work, we consider group size of 3, although larger sizes can also be used. We present inner-group interaction first and inter-group interaction later. We also limit horizon to 2. These two measures allow the per-agent evaluation to be reduced to  $5^{12} \approx 2.4 \times 10^8$  possible effects.

**Graphical Modeling.** These effects are evaluated by agents using a CDN. We denote the three agents in a group by  $A$ ,  $B$  and  $C$ . The subnet dependence structure for  $B$  is shown in Fig. 2 (b). The subnets for  $A$  and  $C$  have the same structure. In the subnet, each decision variable  $mv^{x,i}$  has 5 possible values. Each



**Fig. 2.** (a) The 13 neighborhood cells whose rewards are perceivable by the agent. (b) Subnet structure for agent  $B$ . Design parameter  $mv^{A,1}$  denotes first movement decision of agent  $A$  and is a public variable. Performance measure  $ps_B^{A,1}$  denotes position of  $A$  after first movement and is a private variable in  $B$ . The utility variable  $rw_B^{ABC,2}$  denotes reward received by  $B$  after second movement due to interaction with  $A$  and  $C$ .

position variable  $ps^{x,1}$  has five possible values and each position variable  $ps^{x,2}$  has 13 possible values. Each reward variable  $rw_B^{ABC,i}$  is binary. After compilation, its runtime representation is a cluster tree of 8 clusters (see Fig. 5 (a) for an alternative cluster tree). The size of total state space of this cluster tree (the total number of probability potential values) is 619244. For each round of planing, agent  $B$  must process this state space once for each of the  $5^6 = 15625$  joint plans. Agents  $A$  and  $C$  incur the same amount of computation. The entire round of group planing takes 7380 seconds (2 hours and 3 minutes) running in IBM ThinkPad 2GHz Core Duo (Java implementation without runtime optimization).

**Restricting Inner-Group Interaction.** To speed up the computation, we restrict inner-group agent interaction to pairwise. We only allow direct cooperation between  $A$  and  $B$  and between  $B$  and  $C$ . This essentially imposes an

organizational structure  $A - B - C$  for the group. This restriction will not jeopardize the effectiveness of agent cooperation because the number of agents who can cooperate equals the most effective level of cooperation of the environment (see Section 3).

The subnets for agents  $A$  and  $B$  in the resultant CDN are shown in Fig. 3. The subnet for  $C$  is similar to that of  $A$ . From (a), it can be seen that variables

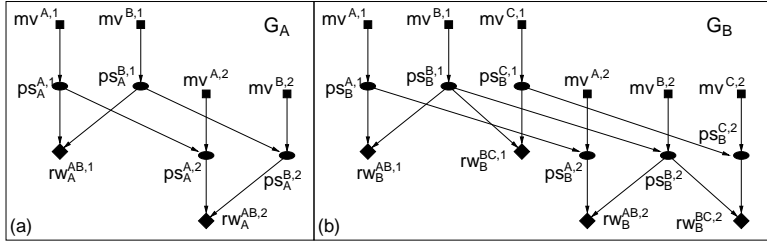


Fig. 3. (a) Subnet for agent  $A$ . (b) Subnet for agent  $B$ .

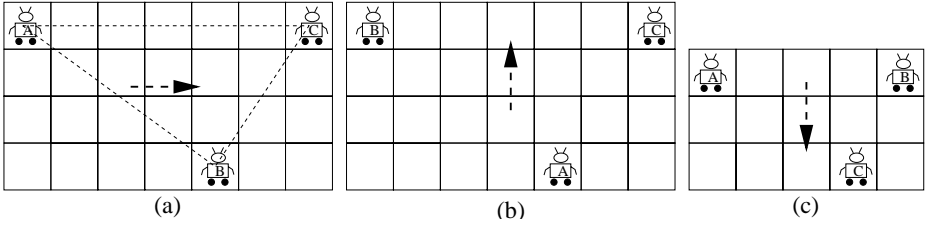
corresponding to agent  $C$  have disappeared from  $G_A$  (the similar occurs with  $G_C$ ). In (b), the rewards due to  $B$ 's cooperations with the other agents have now been decomposed.

The group organization  $A - B - C$  does not, by itself, prevent interactions between  $A$  and  $C$  as they could still meet even though the meeting is not planned. In fact, the group could behave such that all three are trying to meet: an unproductive cooperation. To prevent such behavior, the planning computation steers  $A$  and  $C$  away from each other. How to achieve this is presented below.

**Guiding Agents with Group Direction.** To reduce the state space of agent cluster trees, we require that agents' movements be guided by a *group direction*. If the current group direction is *north*, then  $A$  and  $C$  are not allowed to attempt *south*.

Adopting the group direction allows reduction of the space for variables  $mv^{A,i}$  and  $mv^{C,i}$  by one alternative action, thus reducing the state space of agent cluster trees. Furthermore, this restriction allows the agent group to move less randomly and more strategically because the movements of  $A$  and  $C$  are better coordinated. Note that even though  $A$  and  $C$  are not allowed to attempt *south* in the above scenario, they may still move to south due to the uncertain effect of their movement actions.

Because group direction affects the spaces of public variables  $mv^{A,i}$  and  $mv^{C,i}$ , at any time, the three agents must agree on what is the current group direction. This is achieved by two measures: First of all, group members are required to stay within the 10 step radius to each other. We elaborate later how agents can achieve this through planning with CDN. The consequence is that it allows group members to perceive each other's position. Second, a common algorithm is used by group members to compute the group direction based on their positions.



**Fig. 4.** Determine group directions based on agent positions

The algorithm handles a given situation depending on whether there is a meeting. As mentioned above, the planning computation prevents meeting of the group. Hence, a meeting can occur only between  $A$  and  $B$  or between  $B$  and  $C$ . In such a case, the group direction is pointing from  $A$  to  $C$ . If there is no meeting, the group direction is determined according to the maximum angle in the agent triangle, as illustrated in Fig. 4 (a). Given positions of group members, a triangle is formed with angles,  $\angle A$ ,  $\angle B$  and  $\angle C$ . If a maximum angle exists, it must be either  $\angle B$  as in (a), or  $\angle A$  as in (b), or  $\angle C$  as in (c). The dashed arrow indicates the group direction in each case. If a maximum angle does not exist, a default direction can be used. We omit computational details here due to space limitations.

The method enforces the following properties: First, it steers the group to move in formation  $A - B - C$ . Agent  $B$  is positioned between  $A$  and  $C$ , and three of them tend to arrange into a straight line. This helps prevent unwanted direct interaction between  $A$  and  $C$ . Second, it steers the group to move in the direction pointing from  $A$  to  $C$ . Therefore, the group direction will not change dramatically from move to move, promoting strategic group movement and avoiding wandering around in a small confined region. Third, the group direction does not dictate individual agent movement rigidly. Each agent still has enough flexibility to choose its action. For instance, suppose that bottom right cell in Fig. 4 (c) has a high  $r_2$  value. Then,  $B$  can plan to go *south* twice and  $C$  can plan to *halt* first and then go *east*.

To further reduce the state space of agent cluster trees, we do not allow agents to attempt *halt* in the second movement. Action *halt* is necessary for two agents to meet when they are in certain relative positions such as that of Fig. 4 (c). Our stipulation will force  $C$  to halt in the first step. This requirement, combined with the previous measures, reduces the space of variables  $mv^{A,2}$  and  $mv^{C,2}$  to size 3 and that of  $mv^{B,2}$  to size 4.

The resultant cluster trees for agents  $A$  and  $B$  are shown in Fig. 5. The cluster tree for  $C$  is similar to  $T_A$ . The size of total state space of  $T_B$  is 48169: about 12-fold reduction. The size of total state space of  $T_A$  is 10152: about 61-fold reduction. One round of group planing takes 135 sec (Java implementation without runtime optimization): about a 55-fold speed up.

**Enforcing Desirable Behavior Through Utility.** As mentioned above, within a group, we expect cooperation between  $A$  and  $B$  and between  $B$  and  $C$ , but avoidance of direct interaction between  $A$  and  $C$ . We also expect different

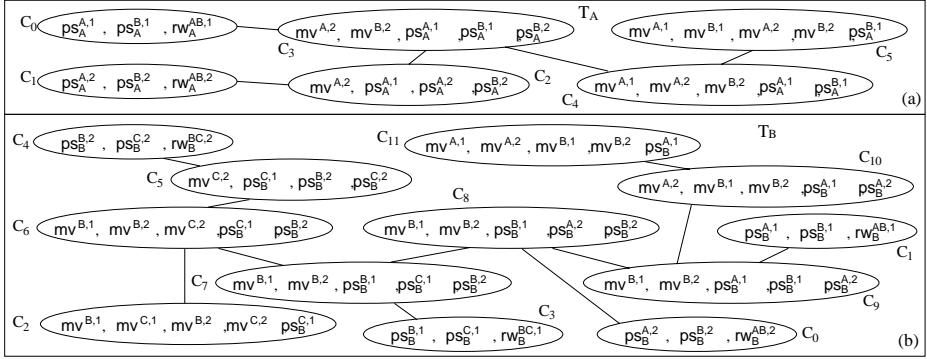


Fig. 5. (a) Cluster tree  $T_A$  for agent A. (b) Cluster tree  $T_B$  for agent B.

groups to avoid each other. We would like to achieve such desirable behavior of agents through reasoning within CDN. To do so, we replace each reward variable  $rw_y^{x,i}$  by a new utility variable  $ut_y^{x,i}$ . Recall that  $rw_y^{x,i}$  is a binary variable and distribution  $P(rw_y^{x,i} = y|\pi(rw_y^{x,i}))$  associated with  $rw_y^{x,i}$  encodes utility function  $u(\pi(rw_y^{x,i}))$ , where  $\pi(rw_y^{x,i})$  is the parent nodes of  $rw_y^{x,i}$  and consists of position variables  $ps_y^{z,i}$ . The distribution  $P(ut_y^{x,i} = y|\pi(rw_y^{x,i}))$  associated with  $ut_y^{x,i}$  is obtained by modifying  $u(\pi(rw_y^{x,i}))$  as follows:

If  $\pi(rw_y^{x,i})$  corresponds to a group configuration where either  $A$  and  $B$  are too far away (according to a distance threshold), or  $B$  and  $C$  are too far away, or  $A$  and  $C$  are too close, or agent  $y$  is too close to any other agent outside the group,  $P(ut_y^{x,i} = y|\pi(rw_y^{x,i}))$  is set to 0. Otherwise,  $P(ut_y^{x,i} = y|\pi(rw_y^{x,i})) = u(\pi(rw_y^{x,i}))$ .

**Jumping in Barren Area.** When a group moves into a *barren* area where all cells have the reward at or below base value, any movement not causing danger will be regarded by the agents just as good as any other. The group movement will then be dominated by danger avoidance and the group could wander around the barren area forever: an unproductive behavior.

To avoid being so trapped, agents could follow an exception rule outside CDN-based planning and move in the group direction for a number of steps. However, such *blind jump*, combined with the uncertain effect of actions, may violate intra and inter-group formation or enter dangerous cells.

We have instead let agents plan their *jump* through CDN. Jumping is triggered by the inference computation with CDN when the optimal group plan has a utility at or below the base value. In the next step, each agent checks the observed rewards for the next two cells in the group direction. If their reward values do not correspond to danger, the cells will be treated as if their reward values are 1 (the highest value) during planning with CDN.

This solution has the following advantages: First, the jumping behavior is produced in the uniform computational framework of normal movement, which



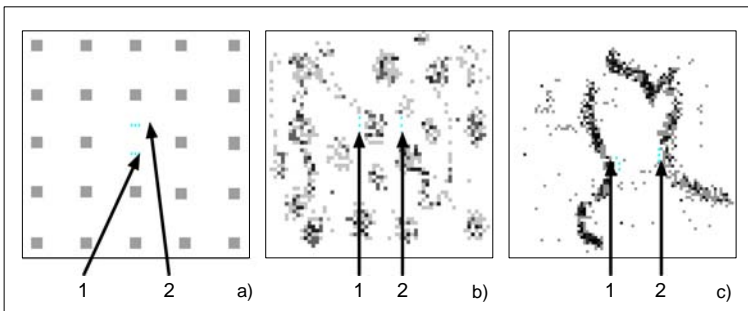
facilitates analysis and implementation. Second, danger avoidance is enabled. Third, if the target cells do not correspond to danger, the high reward values will be used in the distributions associated with variables  $rw_y^{x,i}$ . Because these distributions are subject to the modification into those associated with  $ut_y^{x,i}$ , as mentioned above, desirable behavior regarding intra and inter-group formation will be enforced during jumping. In short, this solution implements jumping along group direction as a soft guideline and combines it with other criteria naturally.

## 5 Experimental Results

To empirically verify the effectiveness of our method, an *Environment Simulator* is implemented as well as the agents. The Simulator models the grid, reward distribution and effect of agent actions (the probability of intended effect of an action is set at 0.9). It feeds agents with observations and updates environmental state according to agent actions. Each agent communicates with group members according to CDN hypertree and with Simulator on observations and action decisions. Agent performance is measured by rewards collected over a finite horizon.

To test the robustness of our method under different environments, three types of reward distributions were used. In a *dense* distribution, every  $10 \times 10$  region has at least one high reward cell. In a *barren* distribution, high reward cells are located in clusters. Each cluster is no larger than a  $6 \times 6$  region and two clusters are at least 20 cells apart. In a *path* distribution, some high reward cells form a pathway and, along the pathway, a high reward cell is no less than two cells away from another one. Examples distributions are shown in Fig. 6. For each distribution type, the initial locations of agents are identical in all runs.

To compare our CDN agents with simpler but non-trivial alternatives, we implemented two types of greedy agents. The domain of decisions  $mv^1$  and  $mv^2$  is  $\{north, south, east, west, halt\}$ . The first type (GRD) is based on unilateral reward  $rw_u$  and maximizes  $rw_u(mv^1) + rw_u(mv^2)$ , where  $rw_u(mv^i)$  is the unilateral reward of the  $i$ 'th movement. The second (GRDB) considers bilateral reward



**Fig. 6.** Example distributions. Left: barren (120 by 120); middle: dense (60 by 60); right: path (80 by 80). Each arrow indicates the starting locations of three agents.

$rw_b$  as well and maximizes  $rw_u(mv^1) + rw_u(mv^2) + rw_b(mv^1) + rw_b(mv^2)$ . For both types, each agent acts independently and there is no direct communication between agents.

Table 1 shows the experimental results. A team of six agents of the same type is used for each run which lasts for 80 time-steps. Each type of agents have five runs for each type of distribution.

**Table 1.** Experimental results. Mean rewards  $\mu$ , as well as max and min, are obtained from 5 runs. Highest means for each distribution is shown as bold.

	CDN			GRD			GRDB		
	$\mu$	max	min	$\mu$	max	min	$\mu$	max	min
barren	<b>51.82</b>	53.9	48.3	49.42	49.8	48.8	49.28	49.8	48.4
dense	<b>72.46</b>	81.5	59.8	60.52	62.4	58.4	68.56	76.5	58.1
path	<b>96.48</b>	100.5	91.1	72.56	78.2	65.2	74.44	84.6	68.7

On average, CDN agents collected highest rewards across all three types of distributions. Results for barren distribution were the closest among the three types of agents. As high reward clusters were far apart, agents' observation range is far below minimal distance between such clusters, and agents were started far from any such clusters, performance of a single run is more subject to chance. As a result, the minimal reward collected by CDN agents were lower than the minimal collected by greedy agents, even though on average CDN agents still outperform. For dense and path distributions, CDN agents outperform greedy agents on every run. For path distribution, the lowest reward collected by CDN agents was higher than the highest reward collected by greedy agents. The superior CDN performance may be attributed to at least two properties. First, CDN agents can plan bilateral visiting of a cell, whereas greedy agents have no direct coordination. Second, CDN agents keep groups apart and thus can explore different regions of the environment and avoid revisiting cells. Of two types of greedy agents, GRDB type outperforms GRD in dense and path distributions since it considered bilateral actions whereas GRD did not.

## 6 Other Related Work

The space precludes an extensive literature review and we discuss only a small subset which is considered the most relevant.

Mazes have been abstracted from office delivery applications and used in empirical study of centralized POMDP algorithms, e.g. [5]. A typical maze consists of walls, hallways, rooms and a single agent. The agent must travel to a goal location through a long sequence of movements. The agent knows the topology of the maze but may not know its starting location. Its sensors can perceive nearby walls but are noisy. In multiagent expedition, at any time, multiple

alternative goals (of different reward) exist for each agent and each requires a short sequence of movements. The objective of planning is to choose among these goals wisely. Agents have no prior knowledge of the environment and the environment is multiagent.

Also abstracting from office delivery applications, Pollack and Ringuette [7] proposed Tileworld multiagent testbed, where agents' goals are to push tiles into holes. As multiagent expedition, a Tileworld agent can pursue one of multiple alternative goals at any time; but unlike multiagent expedition, each goal requires a long sequence of movements. The environment is fully observable (agents can perceive tiles, holes and other agents) and deterministic (actions have intended effects). In contrast, the environment of multiagent expedition is weakly partially observable (agents cannot perceive beyond local region) and stochastic (effects of actions are uncertain).

Tiles and holes in Tileworld dynamically appear and disappear. The feature is reasonable in office delivery applications but insensible in the expedition environment. In multiagent expedition, after being visited by any agent, a cell's reward is reduced to the base value. As a consequence, wandering in the same neighborhood is unproductive and agents must move around strategically.

Work on independent DEC-MDPs [1] shares some features with multiagent expedition. It assumes that actions of one agent cannot affect other's observation and state, and an agent cannot observe other agent's state and communicate with them. In multiagent expedition, agents can observe the state of others if they are close by, they must plan to meet to maximize reward, and CDN utilizes limited inter-agent communication to achieve optimal joint plan.

Noh and Gmytrasiewicz [6] applied the recursive modeling method (RMM) to agents cooperating in anti-missile defense. In their environment, incoming missiles are fully observable. Uncertainty originates from the unknown state of other agents as well as the effect of intercept action.

Artificial birds [8] display formation behavior somewhat similar to the group formation presented in this work. The formation can be viewed as the ends of their birds and the behavior is generated by following simple rules. The formation of agents in expedition is the means to serve the ends. It is generated as part of the desirable behavior through decision-theoretical planning.

Multiagent expedition differs from *exploration*. As commonly referred, e.g., [4,9], the main task of exploration is to produce a map in an unknown environment by moving around and sensing. The map produced can then be used for navigation as in mazes described above. Multiagent expedition, as we presented, does not require a map.

In posing the challenge of Mars rover operations [3], the need to take resource constraints and concurrent actions into account in planning is emphasized. CDNs encode constraints explicitly through design parameters and address concurrent actions through multiagent planning. Hence, CDN based planning provides a promising research direction towards meeting the challenge.

## 7 Conclusion

Multiagent expedition forms a class of DEC-W-POMDPs and captures a number of practical applications. In this work, we solve one instance of DEC-W-POMDP. The knowledge representation is based on CDN, a formal decision theoretic graphical model that supports autonomous, optimal, and efficient multiagent decision making. The generality of CDN allows incorporation of grouping, group direction, jumping and comprehensive preference encoding to achieve efficiency while maintaining optimality. Experiment shows superior performance of CDN agents over greedy agents. Our method can easily scale up to a larger number of agents by employing more groups as well as a larger group size. The addition of groups essentially has no impact on complexity. Large group size with the hyperchain group organization maintained will cause increase of complexity linear on the group size.

Theoretical and experimental comparison of our approach to RMM [6] is underway, through which we hope to gain a better understanding of the pros and cons of our tightly-coupled agent architecture and their loosely-coupled one.

## Acknowledgement

Financial support to the first author through Discovery Grant from NSERC, Canada and that to the second author through OGS from MTCU, Ontario are acknowledged.

## References

1. R. Becker, S. Zilberstein, V. Lesser, and C.V. Goldman. Solving transition independent decentralized Markov decision processes. *J. Artificial Intelligence Research*, 22:423–455, 2004.
2. D.S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov decision processes. In *Proc. 16th Conf. on Uncertainty in Artificial Intelligence*, pages 32–37, Stanford, 2000.
3. J. Bresina, R. Dearden, N. Meuleau, S. Ramkrishnan, D. Smith, and R. Washington. Planning under continuous time and resource uncertainty: a challenge for ai. In *Proc. 18th Conf. on Uncertainty in Artificial Intelligence*, pages 77–84, San Francisco, CA, 2002. Morgan Kaufmann.
4. S. Carpin, H. Kenn, and A. Birk. Autonomous mapping in the real robot rescue league. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII, Lecture Notes in Artificial Intelligence (LNAI) 3020*. Springer, 2004.
5. M.L. Littman, A.R. Cassandra, and L.P. Kaelbling. Learning policies for partially observable environments: scaling up. In A. Prieditis and S. Russell, editors, *Proc. 12th Inter. Conf. on Machine Learning*, pages 362–370, San Francisco, CA, 1995. Morgan Kaufmann.
6. S. Noh and P.J. Gmytrasiewicz. Coordination and belief update in a distributed anti-air environment. In *Proc. 31st Annual Hawaii Inter. Conf. on System Sciences*, pages 142–151, 1998.

7. M. Pollack and M. Ringuette. Introducing the Tileworld: experimentally evaluating agent architectures. In T. Dietterich and W. Swartout, editors, *Proc. 8th National Conf. on Artificial Intelligence*, pages 183–189, Menlo Park, CA, 1990. AAAI Press.
8. C.W. Reynolds. Flocks, herds, and schools. In *Computer Graphics, 21, Proc. SIGGRAPH'87*, pages 25–34, 1987.
9. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.
10. Y. Xiang, J. Chen, and A. Deshmukh. A decision-theoretic graphical model for collaborative design on supply chains. In A.Y. Tawfik and S.D. Goodwin, editors, *Advances in Artificial Intelligence, LNAI 3060*, pages 355–369. Springer, 2004.
11. Y. Xiang, J. Chen, and W.S. Havens. Optimal design in collaborative design network. In *Proc. 4th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'05)*, pages 241–248, 2005.

# Hierarchical Shortest Pathfinding Applied to Route-Planning for Wheelchair Users

Suling Yang and Alan K. Mackworth

Department of Computer Science, University of British Columbia  
Vancouver, British Columbia, Canada V6T 1Z4  
{sulingy,mack}@cs.ubc.ca

**Abstract.** Pathfinding on large maps is time-consuming. Classical search algorithms such as Dijkstra's and A\* algorithms may solve difficult problems in polynomial time. However, in real-world pathfinding examples where the search space increases dramatically, these algorithms are not appropriate. Hierarchical pathfinding algorithms that provide abstract plans of future routing, such as HPA\* and PRA\*, have been explored by previous researchers based on classical ones. Although the two hierarchical algorithms show improvement in efficiency, they only obtain near optimal solutions. In this paper, we introduce the Hierarchical Shortest Path algorithm (HSP) and a hybrid of the HSP and A\* (HSPA\*) algorithms, which find optimal solutions in logarithmic time for numerous examples. Our empirical study shows that HSP and HSPA\* are superior to the classical algorithms on realistic examples, and our experimental results illustrate the efficiency of the two algorithms. We also demonstrate their applicability by providing an overview of our Route Planner project that applies the two algorithms proposed in this paper.

## 1 Introduction

The population of wheelchair users is very significant and increasing dramatically [13]. Therefore, finding accessible wheelchair routes is an important problem. In developed countries, most buildings and public transportation are accessible, making the lives of people in wheelchairs easier. However, the routes to the closest elevator in a new building, temporary road conditions and bus schedules may be unknown to wheelchair users. Therefore, we were motivated to create route-planning software that can be installed on a small device to give wheelchair users route accessibility information while they are travelling. Besides a route planner, our software contains a simple scheduler that synchronizes with the route planner to provide more accurate commuting information for clients. After obtaining the destination from the scheduler, the route planner establishes some possible paths and displays the best one to the client.

The first stage of our project is to implement an algorithm to hierarchically find paths and obtain multiple levels of detail. Since an abstract high-level path can provide a general plan, the client can have an impression of future routing. Instead of being presented with a cumbersome and lengthy low-level path, people, especially the elderly, would prefer a cognitively visible path. In the next

stage, the software accommodates our scheduler and real-world maps, including indoor and outdoor applications.

### 1.1 Problem Statement

The notation used in this section is described here. A map is represented by a graph  $G = (N, E, l)$ , where  $N$  is the set of nodes,  $E$  is the set of edges, and  $l$  is the number of hierarchical levels. Hierarchical levels denote inclusion (containing relation). For example, a building is an ancestor of rooms in it. The cardinalities of  $N$  and  $E$  are denoted by  $n$  and  $e$ . Each *node*  $\in N$  contains a set of neighbours  $neigh(node)$  and is assigned to a level. Each edge  $(i, j) \in E$  is associated with another pair of nodes (*exit*, *entrance*) where *exit* and *entrance* are nodes of one level lower, or they may be *null*. The weight on an edge  $(i, j)$  is denoted by  $w(i, j)$ , and the weight of a path  $P$  is  $w(P) = \sum_{(i,j) \in P} w(i, j)$ . We want to find the *shortest path*  $P$  that has the minimum weight  $w(P)$  from a node  $s$  to another node  $d$  in a graph  $G$ .

**Definition 1: Search** on a graph  $G$ : The process of finding a path  $P = \{s, i, \dots, d\}$ , a sequence of nodes on  $G$ , from the start  $s$  to the destination  $d$ .

Human beings may approach complex problems by dividing them into easier sub-problems, each of which can be further divided into smaller problems or solved by a quicker search. More than forty years ago, Minsky realized that a successful division of a complex problem will greatly reduce the search time by a *fractional exponent* [1]. Such divisions can be considered as “islands” in the search space, where these islands can be abstracted from groups of nodes. In this section, an abstraction of a group of nodes is found when these nodes are at the same level and within the same enclosure. The abstraction then is their enclosure. For example, a building is an abstraction of the rooms in it. Conversely, details at the room level constitute the refinement of the building.

**Definition 2: Hierarchical Pathfinding:** The process of finding a sequence of paths  $\{P_m, P_{m-1}, \dots, P_0\}$ ,  $m \leq l$ , where  $P_i$  contains nodes of  $i$ -th level and that are an abstraction of nodes on  $P_{i-1}$ .

The maximum length of an abstract path then is defined as  $maxlength(P_i) = w(P_i) + \sum_{node \in P_i} upperbound(node)$ , and the minimum length of the path is  $minlength(P_i) = w(P_i) + \sum_{node \in P_i} lowerbound(node)$ , where  $upperbound(node)$  is the maximum of shortest distances between any pair of places within the node and  $lowerbound(node)$  is the minimum distance from the entrance to the exit. The upper bound of a node can be overestimated, whereas the lower bound can be underestimated. Hence,  $upperbound(node)$  can be the total of low-level edge lengths in an abstracted node, *node*, and  $lowerbound(node)$  can be zero.

## 1.2 Previous Work

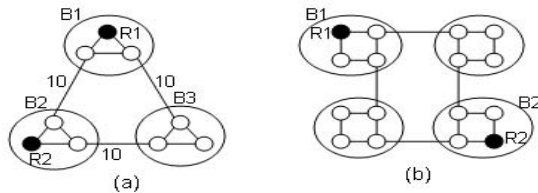
One of the well-known search algorithms is Dijkstra's, a non-heuristic version of A\*. However, it is not as efficient as some A\* algorithms with good heuristics. A\* [5] and IDA\* [6] are both heuristic and complete algorithms on locally finite graphs (graphs with a finite branching factor), but neither of these two algorithms is well-suited into a dynamic environment.

Hierarchical pathfinding has been explored since the mid-nineties, and several excellent algorithms have been developed. In [4], Rabin provides a high-level description on path-finding using a two-level hierarchy. This algorithm uses only two levels of abstraction, but real world maps are divided into numerous levels. In [3], Holte *et al.* explain how abstraction could lead to speedup on finding a solution for a graph-oriented problem. Similar to their work, Hierarchical Pathfinding A\* (HPA\*) [9] and Partial-Refinement A\* (PRA\*) [8] both construct multi-level abstractions using grid-based representation, and greatly reduce the amount of time required to find a near-optimal solution. Instead of estimating a near-optimal solution, our algorithm finds all possible candidates and keeps them until some are proven to be non-optimal.

If the number of sub-level nodes of each node is fixed, the size of the map increases exponentially as the number of levels increases. The running times of existing search algorithms grow exponentially as well, so they may not be practical in reality. In this section, we propose the Hierarchical Shortest Path (HSP) algorithm, as well as a hybrid of the HSP and A\* (HSPA\*) algorithms. HSP finds a threshold that is the least upper-bound of the length of a high-level path, and refines the paths that may be shorter than this threshold. It stops refining a path when the path length exceeds the threshold, and finally returns the shortest path among those that remain. HSPA\* works in a similar way, except that it uses A\* for refinement as soon as it reaches a specific level. The experimental results show that HSP and HSPA\* are suitable for various applications.

## 2 Hierarchical Shortest Path

The hierarchical approach taken in this section involves multi-level representation of a real-life map. For example, these levels can be campuses, buildings,



**Fig. 1.** Examples: (a) and (b) are maps of campuses comprising three and four buildings, respectively, where each building contains a number of rooms



floors and rooms. Assume that a person wants to travel from room  $R_1$  in building  $B_1$  to room  $R_2$  in building  $B_2$  in the world of Fig. 1 (a). If the person begins his pathfinding from  $B_1$  to  $B_2$ , then he could easily find the shortest path from  $B_1$  to  $B_2$  with one edge. If the walking distances within these buildings are negligible, then the person can take the shortest path between  $B_1$  and  $B_2$ , and needs only to refine the path from  $R_1$  to the exit of  $B_1$  and the path from the entrance of  $B_2$  to  $R_2$ . Hence, the person does not need to search paths within  $B_3$ .

Therefore, in order to save time in planning, we can search for short paths between high-level sites first, and then refine the paths within the selected sites for more details. However, the high-level shortest path does not guarantee the shortest overall path, because the paths within each site can vary in length. Consider the previous example, where the walking distance within buildings  $B_1$  and  $B_2$  is significant. Then, a path from  $R_1$  to the exit of  $B_1$ , and a path from the entrance of  $B_2$  to  $R_2$ , may increase the length of the overall path by a large amount. As a result, we can prune away a path  $P$  only when it is guaranteed to be non-optimal, i.e., the minimum length of the path exceeds the maximum length of another path:

$$\exists P' : \text{minlength}(P) > \text{maxlength}(P'). \quad (1)$$

## 2.1 The Algorithm

The main algorithm, 1.1, consists of four major steps: first, find the topmost level abstract nodes wherein the start node and the destination node are different, and abstract the graph  $G$  to that level (line:1,2); second, find a high-level shortest path between ancestors of  $s$  and  $d$  at the  $\hat{l}$ -th level, based on which a threshold is obtained (line:3,4); third, find all possible high-level paths with minimum lengths less than the threshold (line:5); finally, refine each high-level path to the lowest level and return the one with minimum length (line:6,7). This algorithm is complete since it maintains all possible paths and ignores a path only when it is definitely longer than another possible path. The following will explain the algorithm in further detail.

---

### Algorithm 1.1. main ( $G, s, d$ )

---

**Input:** A graph  $G = (N, E, l)$ , the start node  $s$  and the destination node  $d$

**Output:** A shortest path  $P$  from  $s$  to  $d$

- 1:  $\hat{l} := \text{sameAncestorLevel}(s, d) - 1$
  - 2:  $\hat{G} := \text{abstract}(G, \hat{l})$
  - 3:  $\hat{P} := A^*(\hat{G}, \text{ancestor}(s, \hat{l}), \text{ancestor}(d, \hat{l}))$
  - 4:  $\text{threshold} := \text{maxlength}(\hat{P})$
  - 5:  $S := \text{possiblePaths}(\hat{G}, \text{ancestor}(s, \hat{l}), \text{ancestor}(d, \hat{l}), \text{threshold})$
  - 6:  $\hat{S} := \text{refine}(S, l, \text{threshold})$
  - 7: **Return**  $P := \min\{\hat{S}\}$
-

*Abstract of a Graph.* As mentioned above, the desired level in the first step is needed for abstracting the graph. For example, if a person travels from a room in a city to another room in another city within a country, the two cities are the topmost places he should consider. In other words, we seek out the first same ancestor of the two places to find the one level below that ancestor that is desired. Algorithm 1.2 presents the procedure for finding the first same ancestor of two places and returning the level of that ancestor. Note that pathfinding between different level architectures is also allowed. After the desired level  $\hat{l}$  is found, the graph  $G$  is abstracted to that level:

$$\hat{G} := (\hat{N}, \hat{E}), \quad (2)$$

where  $\hat{N}$  and  $\hat{E}$  are subsets of  $N$  and  $E$ , respectively, and the nodes  $node \in \hat{N}$  and  $i, j : (i, j) \in \hat{E}$  are only those at level  $\hat{l}$ .

---

**Algorithm 1.2. sameAncestorLevel** ( $s, d$ )

---

**Input:** The start node  $s$  and the destination node  $d$   
**Output:** The level  $l'$  of the first same ancestor of  $s$  and  $d$   
 1:  $lower = lowerLevelNode(s, d)$   
 2:  $higher = higherLevelNode(s, d)$   
 3: **while**  $lower$  is not at the same level as  $higher$  **do**  
 4:    $lower = parent(lower)$   
 5: **end while**  
 6: **while**  $lower$  is not a sibling of  $higher$  **do**  
 7:    $lower = parent(lower)$   
 8:    $higher = parent(higher)$   
 9: **end while**  
 10: **Return**  $l' := level(lower)$

---

*A High-level Shortest Path and a Threshold.* Either Dijkstra's or the A\* algorithm is used on  $\hat{G}$  to find the shortest path  $\hat{P}$  between the ancestors of  $s$  and  $d$  at the  $\hat{l}$ -th level. Then,  $maxlength(\hat{P})$  is the threshold that we are looking for. The time required to find the high-level path and the threshold is insignificant if the number of high-level nodes is relatively much smaller than that of low-level nodes.

*High-level Possible Paths.* After a threshold is obtained, we could search on  $\hat{G}$  for all possible high-level paths and prune away those with minimum lengths exceeding the threshold. This process is shown in Algorithm 1.3.

*Hierarchical Refinement on High-level Short Paths.* After high-level paths are found, a refinement step is executed, as shown in Algorithm 1.4. It refines each possible high-level path to one level lower each time by recursively finding the shortest path from the entrance to the exit of each node on the high-level path. Then, the lowest level path is constructed by concatenating all partially refined paths. The refinement step ceases if the length of the current path exceeds the threshold.

---

**Algorithm 1.3.** possiblePaths ( $\hat{G}$ ,  $\text{ancestor}(s, \hat{l})$ ,  $\text{ancestor}(d, \hat{l})$ )

---

**Input:** The abstract graph  $\hat{G}$ , the ancestors of  $s$  and  $d$  at level  $\hat{l}$ ,  
and the threshold  $t$

**Output:** A set of paths at level  $\hat{l}$  with minimum lengths less than  $t$

```

1: CurrentPaths  $\leftarrow \{\{\text{ancestor}(s, \hat{l})\}\}$ 
2: Output  $\leftarrow \{\}$ 
3: while CurrentPaths changes do
4:   for each  $P \in \text{CurrentPaths}$  do
5:     if last node  $ln$  of  $P$  is  $\text{ancestor}(d, \hat{l})$  then
6:       Output  $\leftarrow \text{Output} \cup \{P\}$ 
7:       PossiblePaths  $\leftarrow \text{PossiblePaths} \cup \{P\}$ 
8:     else
9:       for each  $nb \in \text{neigh}(ln)$  do
10:         $\hat{P} = P \cup \{nb\}$ 
11:        if  $\text{minlength}(\hat{P})$  is smaller than  $t$  then
12:          PossiblePaths  $\leftarrow \text{PossiblePaths} \cup \{\hat{P}\}$ 
13:        end if
14:      end for
15:    end if
16:  end for
17:  CurrentPaths  $\leftarrow \text{PossiblePaths}$ 
18: end while
19: Return Output

```

---

## 2.2 A Hybrid of HSP and A\* (HSPA\*)

Note that there may be more than one high-level path of shorter length than the threshold, but that is unusual in many realistic cases. The refinement step is fast, if the number of possible short paths is small. Otherwise, the time spent on refining these paths may be excessive. Hence, we could refine our algorithm to select between HSP and A\* depending on the number ( $\alpha$ ) of these paths. Furthermore, from the experimental results shown in the next section, it is evident that HSP is not as fast as A\* if the number of levels is small and the number of sub-nodes is not significantly large. Therefore, we could use A\* algorithm when a high-level path is refined to a specific low-level ( $\beta$ ).

## 2.3 Analysis of Running Time

Let  $b$  be the number of sub-nodes within an abstract node. If  $b$  is fixed, then the total number of lowest-level nodes is  $n = b^l$ , where  $l$  is the number of levels. A\* and IDA\* with good heuristics can solve difficult problems in polynomial time [2,7], i.e.,  $\mathcal{O}(n^c) = \mathcal{O}(b^{cl})$  where  $c$  is a constant. Therefore, the algorithms are exponential to the order of  $l$ . On the other hand, HSP only executes searches on  $\mathcal{O}(b)$  nodes on each level if there is only one abstract path found. In this case, HSP has a running time of  $\mathcal{O}(b^c l)$ . Even if there is more than one abstract path, say  $d$  paths, the running time of HSP is  $\mathcal{O}(b^c d^l)$ . As long as  $d \ll b$ , HSP is still much more efficient than A\* or IDA\*. In reality, where distances within a site

**Algorithm 1.4. refinement** ( $S, l, t$ )**Input:** A set  $S$  of high-level paths, the lowest level  $l$  and the threshold  $t$ **Output:** A set  $\hat{S}$  of low-level paths at level  $l$ 


---

```

1:  $\hat{S} = \{ \}$ 
2: for each hierarchical high-level path  $P \in S$  do
3:    $\hat{P} = \{s\}$ 
4:   for each node  $\in P$  do
5:     find the edge from last node of  $\hat{P}$  to the entrance of node
6:     if  $\text{level}(\text{entrance}) \geq l$  then
7:        $P' = \text{HSP}(\text{entrance}, \text{exit})$  where entrance and exit are within node
8:        $\hat{P} = \hat{P} \cup P'$ 
9:     end if
10:    if  $\text{minlength}(\hat{P}) > t$  then
11:      ignore  $\hat{P}$  and continue the outer loop
12:    end if
13:  end for
14:   $\hat{S} = \hat{S} \cup \{\hat{P}\}$ 
15: end for
16: Return  $\hat{S}$ 

```

---

are usually considered insignificant, one high-level abstract path is expected to appear. Hence, the running time is  $\mathcal{O}(b^c l)$  for the HSP algorithm in real-world examples versus  $\mathcal{O}(b^{cl})$  for A\* and IDA\*, i.e., linear in  $l$  versus exponential in  $l$ . Moreover,  $l$  is usually small while  $b$  is large. Thus,  $\mathcal{O}(b^c l)$  or  $\mathcal{O}(b^c d^l)$  should be relative smaller than  $\mathcal{O}(b^{cl})$ .

## 2.4 Experiments and Results

In real life, each site is connected to its neighbours from exits to entrances. For example, if two buildings are adjacent, then we can exit from a door of one building, and enter through a door of the other building. Hierarchical maps in our experiments are constructed with this connection in mind. Weights of the edges and the upper bound of the shortest distances between any pair of places within an site are included in the map, which is created in XML form. One of the reasons that we use XML form is due to its consistency and flexibility; i.e., you have to define each element with an opening and an ending tag, while you can use an arbitrary tag name. Another reason is that nested tags in XML can represent hierarchical relations easily.

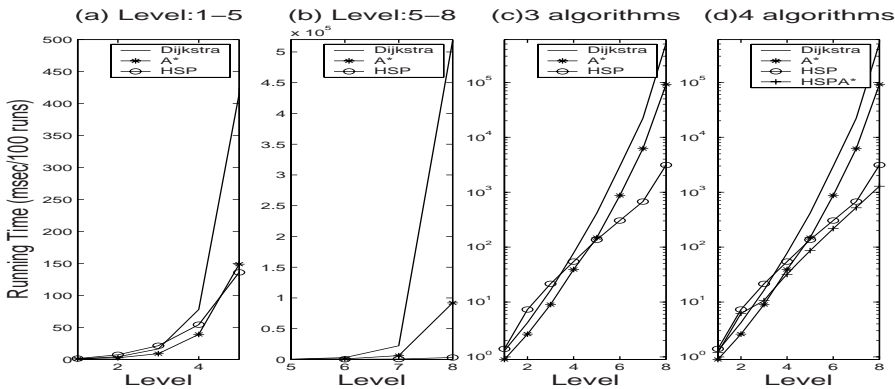
All experiments described in this section were run on a Linux Intel(R) Pentium 4 machine running at 3.20GHz with 2 GB of RAM, using Eclipse version 3.1.

**Examples.** First, we analyze the algorithms using the example shown in Fig. 1 (a). Each building contains three floors, while each floor contains three rooms, and so on. In reality, the number of nodes at one level lower is usually much more than three, where the time saved from pruning away unnecessary high-level

paths is more obvious. Here we use this example because of its simplicity in building a XML file and its simulation of a simple world. The total number of lowest level nodes is  $n_0 = 3^l$ , and since it grows exponentially to the order of  $l$ , Dijkstra's and A\* algorithms are inefficient here. The A\* heuristic used in our experiments is the Euclidean distance, which is popular to use in real-world route-finding problems. However, this heuristic may not be useful for a multilevel structure, since the heuristic values for low-level nodes may be similar. For example, the Euclidean distances from nearby rooms in one city to a room in another city may be indistinguishable. In contrast, HSP, which prunes away most insignificant high-level paths, runs well in these examples.

As shown in Fig. 2 (a) and (b), both multi-level representation running times of Dijkstra's and A\* algorithms blow up quickly, while the HSP running time grows slowly. A semi-log graph of the three algorithms' running times is drawn in Fig. 1 (c). We can observe that the slope of the HSP curve is significantly less steep than that of the other two curves. A further observation from Fig. 2 (c) is that HSP is not superior among the three algorithms all the time. When the number of levels is small, HSP wastes time on recursive calls. Therefore, it is better to use a hybrid of HSP and A\*, the HSPA\* (as described in Sect. 2.2). The running-time results for HSPA\* are compared with those of the other three algorithms in Fig. 2 (d). Note that HSP and HSPA\* have similar performances in these examples, although HSPA\* is more stable.

Table 1 shows the running times of the four algorithms in the example where every abstract node has four sub-nodes, as shown in Fig. 2 (b). In this example, more than one abstract high-level path is found at each level, but the running times of HSP and HSPA\* are still promisingly good. Examples with larger number of sub-nodes ( $b$ ) are also explored, and the outcomes are shown in Table 1 as well.



**Fig. 2.** Running time plots: (a), (b): the running times of three algorithms; (c): the running times of three algorithms in log-scale; (d): the running times of four algorithms in log-scale

**Table 1.** Running times: The running times (in milliseconds) of four algorithms using examples where every abstract node has four or more sub-nodes ( $b$ )

$(b) :$	(4)				(5)			
Level:	1	2	3	4	1	2	3	4
Dijkstra	2	6	35	295	2	11	94	1278
A*	1	4	22	149	1	10	56	717
HSP	1	8	23	130	2	12	42	97
HSPA*	1	8	24	174	1	11	35	88
$(b) :$	(6)				(7)			
Level:	1	2	3	4	1	2	3	4
Dijkstra	3	24	350	7817	3	25	466	15601
A*	2	23	218	3462	2	21	231	8368
HSP	2	28	131	354	2	31	137	368
HSPA*	2	24	99	296	2	28	116	316

## 2.5 Application

Several existing projects, such as the Assistant Cognition and the Aphasia projects, have been helping physically challenged people to better perform daily activities [10,12]. The usefulness of these projects motivates us to build a cognitive assisted system for people with disabilities.

*Route Planner for People with Disabilities.* Our route planner, which shows accessible routes for wheelchair users contains the data of several buildings at the University of British Columbia (UBC): ICICS, the X building and Dempster, the three main buildings in our department. This system is based on a scheduler system, which can be Microsoft Outlook, ESI Planner II [11], or others. We used Microsoft Outlook for our system because of its ease of use and compatibility with other systems. Based on an accurate schedule, the destination can be easily estimated. Then, the route planner computes the possible paths to the destination using HSPA\* and shows both high-level path and low-level paths to the client. The system is installed on a small device, such as a tablet PC, so that users can carry the device and find a path to destination while they are travelling. For example, if a client is heading to class, then Fig. 3 shows a high-level path from ICICS to Dempster, where most classes are held. After the user acquires an overview of the whole path, he/she clicks the “Detail” button. Then, the low-level path corresponding to the high-level path is shown, as in Fig. 4. (See Appendix A for more details.)

## 3 Conclusion and Future Work

In this paper, the Hierarchical Shortest Pathfinding (HSP) algorithm is presented and its running time analyzed. The speedup from eliminating unnecessary

high-level paths is remarkable, and good performances of HSP is expected in real-world pathfinding problems. There are a few directions that this research could be extended beyond the work shown in this paper. First, more examples are anticipated to analyze the performances of the four algorithms. We analyzed the running times of the HSP and HSPA\* algorithms based on artificial examples. We shall apply the algorithms on more actual examples, such as the whole UBC campus. Second, as described in Sect. 2.2, the HSPA\* algorithm contains two parameters, namely  $\alpha$  and  $\beta$ . The best-fit values of these two parameters may vary depending on different problems. Therefore, it may be worthwhile learning well-suited values for these two parameters using stochastic local search methods. Third, comparison of HSP, HSPA\* and other hierarchical pathfinding algorithms could be further investigated in terms of their running times and performance.

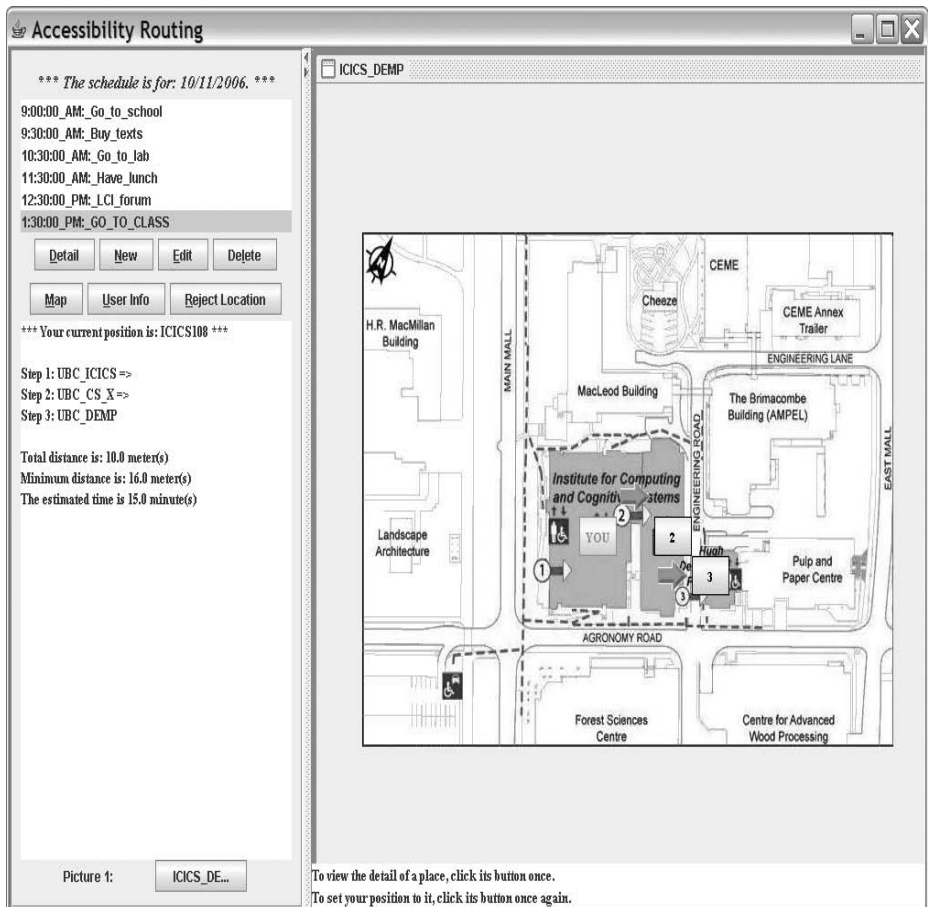
## References

1. Minsky, M.: Steps toward artificial intelligence. (1961)
2. Russell, S., Norvig, P.: Solving problems by searching. *Artificial Intelligence: A Modern Approach*. (1995)
3. Holte, R., Mkadmi, T., Zimmer, R. M., MacDonald, A. J.: Speeding up problem solving by abstraction: A graph oriented approach. *Artificial Intelligence*. (1996) 85(1-2):321-361
4. Rabin, S.: A\* aesthetic optimizations. *Game Programming Gems*. (2000) 264-271
5. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems Science and Cybern.* (1968) 4 100-107
6. Korf, R.: Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*. (1985) 27(1):97-109
7. Korf, R., Reid, M., Edelkamp, S.: Time complexity of iterative deepening-A\*. *Artificial Intelligence*. (2001) 199-218
8. Sturtevant, N., Buro, M.: Partial pathfinding using map abstraction and refinement. *AAAI-05*. (2005) 1392-1397
9. Botea, A., Müller, M., Schaeffer, J.: Near optimal hierarchical path-finding. *J. of Game Develop.* (2004) 7-28
10. Kautz, H., Fox, D., Etzioni, O., Borriello, G., Arnstein, L.: An overview of the assisted cognition project. *American Association for Artificial Intelligence*. (2002)
11. Moffatt, K., McGrenere, J., Purves, B., Klawe, M.: The participatory design of a sound and image enhanced daily planner for people with aphasia. *Proceedings of ACM CHI*. (2005) 501-510
12. McGrenere, J., Davies, R., Findlater, L., Graf, P., Klawe, M., Moffatt, K., Purves, B., Yang, S.: Insights from the aphasia project. *Proceedings of ACM Conference on Universal Usability*. (2003) 112-118
13. Wheelchair Foundation: <http://wheelchairfoundation.ca/> (2002)

## A Screen Shots

Users can select one of the events from the event list on the left hand side of the main frame to view the event information. If a path is found for that event, then

the high-level path is shown, as in Fig. 4. Note that the buttons that represent the corresponding (sub)destinations are located where the (sub)destinations are. The “YOU” button represents the current position of the client. Clicking on one of the buttons that represents (sub)destinations will display useful information regarding the desired (sub)destination on the left-hand pane. The whole path with specific steps (nodes on the path) is shown on the left-hand as well. Since nodes are not able to show on the same picture, buttons for each picture should be added so that users can view any portion or the whole path. These buttons are placed on the left bottom corner, as shown in Fig. 3 to 4. To view the detailed path, the user can click the “Detail” button, and then Fig. 4 will show.



**Fig. 3.** A screen shot of the Route Planner when a high level path is shown



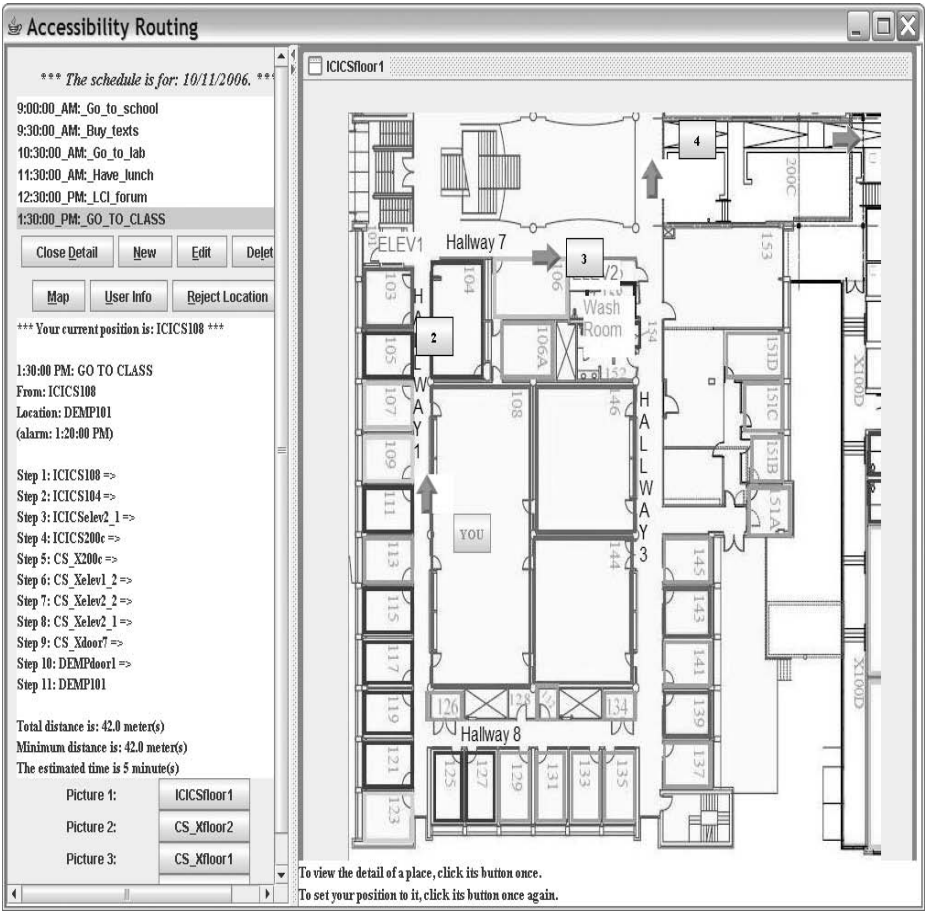


Fig. 4. A screen shot of the Route Planner when a low level path is shown

# Author Index

- Abdullah, Nabil 502  
Avoli, Massimo 192  
Aykut, Murat 122  
  
Beattie, Brien 204  
Benferhat, Salem 356  
Bergler, Sabine 476, 489  
Besse, Camille 50  
Boerlage, Brent 344  
Boularias, Abdeslam 26  
Buffett, Scott 369  
Burkov, Andriy 26  
  
Cai, Zhihua 171  
Calitoiu, Dragos 320  
Carrasco-Ochoa, J.Ariel 146  
Chaib-draa, Brahim 26, 50  
Chen, Shaoju 369  
Chen, Yaohua 296  
Couturier, Olivier 393  
Crowley, Mark 344  
  
de Guzman, Philip 192  
Deng, Weihong 241  
Dou, Dejing 272  
Druzdzal, Marek J. 332  
Du, Jun 171  
Dudek, Gregory 417  
Duin, Robert P.W. 110  
  
Ekinci, Murat 122  
  
Feiguina, Olga 452  
Fleming, Michael W. 369  
Fournier-Viger, Philippe 393  
Frost, Richard A. 502  
  
Gerhard, David 204  
Giguère, Philippe 417  
Gopalan, N.P. 250  
Gu, Baohua 284  
Guo, Jun 241  
  
Hacid, Hakim 405  
Haenni, Rolf 464  
Hanshar, Frank 526  
He, Liu 381  
  
Heinemann, Bernhard 308  
Hernández-Rodríguez, Selene 146  
Hilderman, Robert J. 204  
Hu, Jiani 241  
  
Ji, Ae-Ttie 14  
Jo, Geun-Sik 14  
  
Karnick, Harish 441  
Kashani, Mehdi M. 284  
Kim, Heung-Nam 14  
Kim, Jindae 38  
Kim, Jongwan 272  
Kim, Kwangsoo 86  
Kim, Sang-Woon 110  
Krestel, Ralf 489  
Kumara, Soundar R.T. 38  
Kwak, Donghwi 272  
  
Lane, Terran 429  
Lapalme, Guy 159, 452  
Ling, Charles X. 171  
Liu, Haishan 272  
Liu, Yudong 284  
  
Mackworth, Alan K. 539  
Marinakakis, Dimitri 417  
Martínez-Trinidad, J. Francisco 146  
Melli, Gabor 284  
Mellouli, Sehl 61  
Mephu Nguifo, Engelbert 393  
Mouhoub, Malek 216  
  
Nathan, Michel 73  
Nicolai, Garrett 204  
Nkambou, Roger 393  
Nussbaum, Doron 320  
  
Ok, Changsoo 38  
Oommen, John B. 320  
  
Peng, Yun 98  
Pineau, Joelle 192  
Plamondon, Pierrick 50  
Poole, David 344  
Popowich, Fred 284

- Ridens, Martin 429  
Ryoo, Hong Seo 86
- Saradhi, V. Vijaya 441  
Sarkar, Anoop 284  
Shi, Zhongmin 284  
SivaSelvan, B. 250  
Smaoui, Salma 356  
Sokolova, Marina 159  
Stevens, Scott 429  
Sukpan, Amrudee 216
- Vincent, Robert D. 192  
Viswanathan, Murlikrishna 261  
Voll, Kimberly 514
- Wachter, Michael 464  
Wang, Bin 180  
Wang, Yang 284
- Witte, René 476, 489  
Wu, Dan 381
- Xiang, Yang 228, 526
- Yan, Mingwu 134  
Yang, Suling 539  
Yao, Yiyu 134, 296  
Yee, Shang-Tae 38  
Yeon, Cheol 14  
Yoshida, Tetsuya 405  
Yu, Yang 98  
Yuan, Changhe 332
- Zhang, Harry 180  
Zhang, Wanling 228  
Zhang, Yu 1  
Zhao, Yan 134, 296